Software License Agreement

This is a legal agreement between you, the end user, and Scanalytics. The term “SOFTWARE” as used in this license agreement includes the IPLab for Macintosh program, Camera Controls, Optional Extensions, Libraries, Scripts and all accompanying documentation. CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS BEFORE USING THIS SOFTWARE, OPENING THIS PACKAGE OF THE SOFTWARE, OR ANY DOWNLOADING, REPRODUCTION, COPYING, OR OTHER USE OF THE SOFTWARE BY YOU WILL CONSTITUTE ACCEPTANCE BY YOU OF THE TERMS AND CONDITIONS HEREIN.

LICENSE

You, the user, acknowledge that the SOFTWARE is the sole and exclusive property of Scanalytics, Incorporated. You have the right to use the SOFTWARE as outlined in, and limited by, this agreement. This SOFTWARE may only be used on a single terminal connected to a single computer (i.e. CPU). You may transfer the SOFTWARE from one computer to another provided that the SOFTWARE is used only on one computer at one time. You may not modify, translate, adapt, reverse engineer, decompile, disassemble, or create derivative works based upon this SOFTWARE. You agree to use your best efforts and take all reasonable steps to protect all aspects of the SOFTWARE from unauthorized reproduction, publication, disclosure, or distribution.

COPYRIGHT

Scanalytics, Incorporated owns all rights regarding the SOFTWARE, including the copyright. The SOFTWARE is valuable property of Scanalytics, Incorporated and the look and feel of the SOFTWARE are protected by the copyright laws of the United States and other countries. Under the copyright laws, the SOFTWARE, manuals, and product designs may not be copied, in whole or in part, via any means mechanical, electronic or otherwise, without the written consent of Scanalytics, Incorporated, except to make backup copies of the SOFTWARE, or to transfer the SOFTWARE to a hard disk. This exception does not allow copies to be made for other users. Under the law, copying includes translation to any other language or format. Unauthorized use of or copying of the SOFTWARE or the accompanying written material is an infringement of the copyright belonging to Scanalytics, Incorporated. Such infringement subjects the violator to civil and criminal penalties.

LIMITATIONS ON WARRANTY AND LIABILITY

If physical defects are discovered in the original media on which the SOFTWARE is distributed, Scanalytics, Incorporated will replace the media at no charge to you within a period of 90 days from the date of purchase. You must return the defective item(s) to Scanalytics, Incorporated. Scanalytics, Incorporated will replace the materials free of charge, provided you have previously returned your product registration card to Scanalytics, Incorporated. This warranty will be void if you have in any way modified the SOFTWARE, if you are in breach of this agreement, or if failure of the magnetic media has resulted from accident, abuse, or misapplication. The remedy for breach of this warranty shall be limited to replacement and shall not encompass any other damages. THESE REMEDIES ARE NOT AVAILABLE OUTSIDE OF THE UNITED STATES OF AMERICA. Other than the limited warranty described above, Scanalytics, Incorporated makes no warranty or representation, either express or implied, with respect to this SOFTWARE, its quality, performance, merchantability, or fitness for any particular purpose. In no event will Scanalytics, Incorporated or anyone else involved in the creation, production, or delivery of the SOFTWARE be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the SOFTWARE. In particular, but without limiting the generality of the foregoing, Scanalytics, Incorporated shall have no liability for any damage to programs or data used with Scanalytics, Incorporated products, including the costs of recovering such programs or data. The warranty and remedies set forth above are exclusive and in lieu of all others, oral or written, express or implied. No dealer, agent, or employee of Scanalytics, Incorporated is authorized to make any modification, extension, or addition to this warranty. This warranty gives you specified legal rights. You may have additional rights which vary from state to state.

UPDATE POLICY

In order to obtain updates of these programs, you must complete the registration card and return it to Scanalytics. If Scanalytics has not received your Registration Card, Scanalytics is under no obligation to make any updates available to you. All updates provided to you shall become part of the material governed by the terms of this agreement.

IPLab is a trademark of Scanalytics, Incorporated. Macintosh is a trademark of Apple Computer, Inc.
# Table Of Contents

## 1 Must-Read Section: Getting Started

1.1 What Equipment You Need ................................................................. 1
1.2 Setting Up IPLab .................................................................................. 2
1.3 Allocating More Memory to IPLab .................................................... 3

## 2 Must-Read Section: Quick-start with the IPLab Interface

2.1 #1: The Menu Bar .................................................................................. 6
2.2 #2: The Image Window ........................................................................... 9
  2.2.1 #2a: The Image .................................................................................. 9
  2.2.2 #2b: The Region of Interest (ROI) .................................................... 11
  2.2.3 #2c & 2d: Segment and Drawing Overlays ..................................... 13
2.3 #3-6: The Tools Window ........................................................................ 14
2.4 #3: The ROI Tools ................................................................................ 15
  2.4.1 ROIs in Text Windows ........................................................................ 16
  2.4.2 Moving and Resizing ROIs ............................................................... 16
  2.4.3 Scripting the ROI Tools .................................................................... 16
2.5 #4: The Drawing Tools ........................................................................... 16
  2.5.1 Scripting the Drawing Tools ............................................................. 20
2.6 #5: The Segment Tools .......................................................................... 20
  2.6.1 Scripting the Segment Tools ............................................................. 24
2.7 #6: The Image Tools ............................................................................... 24
  2.7.1 Scripting the Image Tools ................................................................. 26
2.8 #7: The FKeys (Function Keys) Palette .................................................. 26
2.9 #8: The Status Palette .......................................................................... 27
2.10 #9: The Variables Window ................................................................. 28

## 3 Tutorials

3.1 Tutorial 1: Getting Started .................................................................. 29
  3.1.1 Start IPLab and Open an Image ...................................................... 29
  3.1.2 The Status palette .......................................................................... 31
  3.1.3 Using the ROI and Image Tools ...................................................... 32
  3.1.4 Using Segment and Drawing Tools ................................................. 35
3.2 Tutorial 2: Image Enhancement .......................................................... 37
  3.2.1 Overview of the Enhance Menu ...................................................... 37
  3.2.2 Contrast Enhancement .................................................................... 38
  3.2.3 Filtering .......................................................................................... 50
  3.2.4 Color Image Filtering ...................................................................... 51
  3.2.5 Geometric Operations .................................................................... 54
  3.2.6 Adjusting the size of the image ...................................................... 55
  3.2.7 Color Overlaying or Color Merging ................................................ 56
  3.2.8 Other Enhance Commands ............................................................. 57
3.3 Tutorial 3: Particle Segmentation ........................................................ 58
  3.3.1 What is a Segment? ........................................................................ 58
  3.3.2 Hand Drawing ................................................................................ 58
  3.3.3 Placing a Segment at the ROI ......................................................... 60
  3.3.4 Copy and Paste a Segment ............................................................. 61
  3.3.5 Intensity Thresholding ..................................................................... 62
  3.3.6 Cleaning Segments: Binary Morphology ....................................... 64
  3.3.7 Using Grain Boundaries ............................................................... 67
  3.3.8 Pre-Processing For Texture ........................................................... 68
  3.3.9 Segmentation on Color Images ....................................................... 70
  3.3.10 Summary ...................................................................................... 75
4 Should-Read Section ................................................................. 119
  4.1 # / V Buttons, L / SV Buttons .............................................. 119
  4.2 IPLab Variables ..................................................................... 119
    4.2.1 What Are IPLab Variables?.............................................. 119
    4.2.2 Numeric Variables.......................................................... 120
  4.3 Keyboard Keys....................................................................... 121
    4.3.1 Option Key ...................................................................... 121
    4.3.2 Arrow keys ....................................................................... 121
    4.3.3 Command-Period and Escape Key ................................. 122
    4.3.4 Delete Key ....................................................................... 122
    4.3.5 Shift Key ......................................................................... 122
    4.3.6 Quit without Saving: Command-Option-Shift-Quit ........ 122
  4.4 Batch File Processing ............................................................. 123
  4.5 IPLab Text Files ..................................................................... 123
    4.5.1 Importing Text Files......................................................... 123
    4.5.2 Exporting Text Files......................................................... 124
  4.6 Data Views............................................................................ 125
  4.7 Saving IPLab Settings ............................................................. 125
5 Operation ................................................................................... 127
  5.1 File Formats ......................................................................... 127
    5.1.1 Descriptions of File Formats ............................................ 127
    5.1.2 Data Saved in Each File Format ...................................... 128
    5.1.3 Other Formats................................................................. 128
  5.2 Data Types ........................................................................... 129
    5.2.1 Indexed Color ................................................................. 129
    5.2.2 Direct Color ..................................................................... 130
  5.3 Copy-Paste Operations ........................................................... 134
6 Scripting ................................................................. 135
  6.1 What is a Script? ................................................................. 135
  6.2 How to Script ...................................................................... 136
  6.3 Using the Script Window Buttons ........................................ 136
    6.3.1 Record ON/OFF ............................................................ 137
    6.3.2 Script Info Button (i button) .......................................... 137
    6.3.3 Continue and Run Buttons ........................................... 137
    6.3.4 Script Docking button (l button) .................................... 138
  6.4 Recording the Script .......................................................... 138
    6.4.1 Command List ............................................................ 138
    6.4.2 Interactivity: Making Dialogs Open .............................. 139
    6.4.3 Commenting Your Script ............................................. 139
  6.5 Using the Script Commands Palette .................................... 140
    6.5.1 Editing the Script: Cut, Copy, Paste, Clear, Duplicate ...... 140
    6.5.2 Saving and Printing the Script: Save, Save As, Print ....... 140
    6.5.3 Insert ........................................................................ 140
    6.5.4 Revert ........................................................................ 140
    6.5.5 Dialog, Comment ....................................................... 140
    6.5.6 Alert ........................................................................... 141
    6.5.7 Query ......................................................................... 142
    6.5.8 Enter Var (Enter Variables) .......................................... 143
    6.5.9 Label .......................................................................... 144
    6.5.10 If ............................................................................... 145
    6.5.11 Jump .......................................................................... 145
    6.5.12 Loop .......................................................................... 146
    6.5.13 Delay .......................................................................... 146
  6.6 Running, Continuing, and Stopping a Script ......................... 148
  6.7 Running One Script from Another Script ............................. 149
  6.8 Error Handling ................................................................... 150
  6.9 Startup Script ..................................................................... 150
  6.10 IPLab Scripts Folder .......................................................... 150

7 Menu Reference ................................................................. 151
  7.1 Introduction ........................................................................ 151
    7.1.1 Overview of the Menus ................................................ 151
    7.1.2 Default Parameter Settings .......................................... 151
    7.1.3 Interactive vs. Script Operation .................................... 152
    7.1.4 Errors ......................................................................... 152
    7.1.5 Commands Which Operate on Direct Color Images ....... 152
    7.1.6 Analyzing Regions ..................................................... 152
    7.1.7 Apple Events and AppleScript ................................. 153
    7.1.8 Apple Menu ............................................................... 154
7.2 File Menu........................................................................................................................................ 155
  7.2.1 New Image.................................................................................................................................. 155
  7.2.2 Open........................................................................................................................................... 156
  7.2.3 Open As ...................................................................................................................................... 159
  7.2.4 Close Image ............................................................................................................................. 160
  7.2.5 Save........................................................................................................................................... 160
  7.2.6 Save As ...................................................................................................................................... 161
  7.2.7 Revert to Saved ......................................................................................................................... 164
  7.2.8 Delete File.................................................................................................................................. 165
  7.2.9 Export ......................................................................................................................................... 166
  7.2.10 Edit File List ............................................................................................................................ 169
  7.2.11 Select File List .......................................................................................................................... 171
  7.2.12 Set Indexed Name .................................................................................................................. 172
  7.2.13 Foreign Formats ...................................................................................................................... 173
  7.2.14 Edit Image Info ....................................................................................................................... 176
  7.2.15 Define Image Info .................................................................................................................. 177
  7.2.16 Edit FITS Header .................................................................................................................... 177
  7.2.17 Page Setup ............................................................................................................................... 178
  7.2.18 Print .......................................................................................................................................... 179
  7.2.19 Quit .......................................................................................................................................... 180

7.3 Edit Menu....................................................................................................................................... 181
  7.3.1 Undo.......................................................................................................................................... 181
  7.3.2 Cut............................................................................................................................................. 181
  7.3.3 Copy.......................................................................................................................................... 182
  7.3.4 Paste ......................................................................................................................................... 183
  7.3.5 Clear .......................................................................................................................................... 185
  7.3.6 Crop Image .............................................................................................................................. 186
  7.3.7 Set Value ................................................................................................................................... 187
  7.3.8 Set ROI Color .......................................................................................................................... 187
  7.3.9 Select All .................................................................................................................................... 188
  7.3.10 Define ROI ............................................................................................................................. 190
  7.3.11 Modify ROI .............................................................................................................................. 191
  7.3.12 Delete Frames ......................................................................................................................... 192
  7.3.13 Transfer Frames ...................................................................................................................... 192
  7.3.14 Reverse Sequence ................................................................................................................... 193
  7.3.15 Change Drawings .................................................................................................................... 193
  7.3.16 Clear All Drawings ................................................................................................................... 193
  7.3.17 Select Next Drawing .............................................................................................................. 193
  7.3.18 Transfer Attributes ................................................................................................................ 194
  7.3.19 Set Row/Column Label ......................................................................................................... 195
  7.3.20 Stamp Overlays On Image ..................................................................................................... 195
  7.3.21 Assign F Keys ......................................................................................................................... 196
  7.3.22 Preferences ............................................................................................................................. 197

7.4 View Menu..................................................................................................................................... 198
  7.4.1 Show/Hide Image ....................................................................................................................... 198
  7.4.2 Home Image ............................................................................................................................. 198
  7.4.3 Select Frame ............................................................................................................................. 198
  7.4.4 Show/Hide ROI ........................................................................................................................ 199
  7.4.5 Show/Hide Drawing, Show/Hide Segment ............................................................................. 199
  7.4.6 Show/Hide Labels .................................................................................................................... 200
  7.4.7 Photo ......................................................................................................................................... 200
  7.4.8 Animate .................................................................................................................................... 200
  7.4.9 View Options ............................................................................................................................ 201
  7.4.10 View As .................................................................................................................................... 205
7.5 Enhance Menu ........................................................................................................................................207
  7.5.1 Normalization ..................................................................................................................................207
  7.5.2 White Balance ..............................................................................................................................210
  7.5.3 Equalize Contrast ..........................................................................................................................211
  7.5.4 Image Ratios ...................................................................................................................................212
  7.5.5 Invert .............................................................................................................................................213
  7.5.6 Pseudocolor .....................................................................................................................................214
  7.5.7 Edit Color Table .............................................................................................................................215
  7.5.8 Apply Color Table ..........................................................................................................................216
  7.5.9 Color To Grayscale .........................................................................................................................216
  7.5.10 Despeckle .......................................................................................................................................216
  7.5.11 Linear Filter ...................................................................................................................................217
  7.5.12 Median Filter ..................................................................................................................................218
  7.5.13 Edge Filter ......................................................................................................................................218
  7.5.14 FFT Filter .......................................................................................................................................219
  7.5.15 Morphology ....................................................................................................................................221
  7.5.16 Define Linear Filter .........................................................................................................................221
  7.5.17 Flip ..................................................................................................................................................222
  7.5.18 Register ..........................................................................................................................................224
  7.5.19 Rotate & Scale ................................................................................................................................225
  7.5.20 Mosaic .............................................................................................................................................227

7.6 Analyze Menu .........................................................................................................................................229
  7.6.1 Segmentation ....................................................................................................................................229
  7.6.2 Segment at ROI .................................................................................................................................233
  7.6.3 Modify Segments ...............................................................................................................................233
  7.6.4 Change Segment Color ....................................................................................................................234
  7.6.5 Set XY Units .....................................................................................................................................235
  7.6.6 Define XY Units ...............................................................................................................................236
  7.6.7 Set Measurements .............................................................................................................................237
  7.6.8 Measurement Options ......................................................................................................................241
  7.6.9 Measure ROI ....................................................................................................................................241
  7.6.10 Measure Segments .........................................................................................................................242
  7.6.11 Quantify Segments .........................................................................................................................242
  7.6.12 Histogram ......................................................................................................................................243
  7.6.13 Row/Col Plot ..................................................................................................................................244
  7.6.14 Contours .........................................................................................................................................244
  7.6.15 Transforms .....................................................................................................................................245
  7.6.16 Calibrate Intensities .........................................................................................................................247
  7.6.17 Dispose Segment Layer ..................................................................................................................248
  7.6.18 Extract ...........................................................................................................................................249
7.7 Math Menu ............................................................................................................................... 251
  7.7.1 Convert to 8/24 Bit........................................................................................................... 251
  7.7.2 Unsigned<->Signed...................................................................................................... 251
  7.7.3 Change Data Type..................................................................................................... 252
  7.7.4 Change Color Type.................................................................................................... 252
  7.7.5 Subsample .................................................................................................................. 253
  7.7.6 Interleave Zeros .................................................................................................... 254
  7.7.7 Sort Data .................................................................................................................. 254
  7.7.8 Barrel Shift ............................................................................................................. 255
  7.7.9 Swap 0 and 255 ..................................................................................................... 255
  7.7.10 Binarize .................................................................................................................. 256
  7.7.11 Pixel Arithmetic ................................................................................................... 257
  7.7.12 Image Arithmetic .................................................................................................. 258
  7.7.13 Complex ................................................................................................................ 259
  7.7.14 Set Pattern ............................................................................................................ 260
  7.7.15 Encode-Decode ..................................................................................................... 262
  7.7.16 Split Color Channels ........................................................................................... 263
  7.7.17 Merge Color Channels ....................................................................................... 264
  7.7.18 Convert Color Channels ...................................................................................... 266
  7.7.19 Affine Color Transform ....................................................................................... 266

7.8 3D ............................................................................................................................................. 268
  7.8.1 General Description and Applications ...................................................................... 268
  7.8.2 Overview of 3D Visualization .................................................................................. 269
  7.8.3 3D Equalize ............................................................................................................. 269
  7.8.4 3D Filter .................................................................................................................... 271
  7.8.5 3D ROI Sum Plot .................................................................................................. 275
  7.8.6 3D Mosaic .............................................................................................................. 276
  7.8.7 3D Oblique View ................................................................................................... 278
  7.8.8 3D Stacked View ................................................................................................... 279
  7.8.9 3D Tracer ............................................................................................................... 280
  7.8.10 3D Projector ......................................................................................................... 283
  7.8.11 Sample 3D Scripts ............................................................................................... 286

7.9 Camera Menu ........................................................................................................................... 288

7.10 Control Menu ........................................................................................................................... 288

7.11 Script Menu .............................................................................................................................. 289
  7.11.1 New Script ............................................................................................................. 289
  7.11.2 Open Script .......................................................................................................... 289
  7.11.3 Run Script ............................................................................................................. 290
  7.11.4 Set Variable ........................................................................................................... 291
  7.11.5 Action Variables .................................................................................................. 292
  7.11.6 Continue ............................................................................................................... 294
  7.11.7 Cancel All ............................................................................................................. 294
1 Must-Read Section: Getting Started

1.1 What Equipment You Need

The material in this User’s Guide is self contained. Although it is helpful to have an understanding of image processing methods and terminology in order to develop scripts for your own algorithms, it is not necessary for an understanding of the manual.

IPLab for Macintosh uses the standard Macintosh interface. This manual assumes that you are familiar with these standard techniques: clicking the mouse, dragging, using the keyboard, accessing menus, interacting with dialogs, manipulating windows, handling files, and other Macintosh fundamentals. If you are unfamiliar with these techniques, please consult your computer's manual for descriptions of their use.

To use IPLab for Macintosh version 3.6, you need the following:

- Any Macintosh computer running OS 8.0 or later.
- At least 128 megabytes of memory free to allocate to IPLab for Macintosh. Be aware that image processing requires a lot of memory to store the image data, so the more memory you have assigned to IPLab, the better performance you will see.
- A hard disk and a CD-ROM drive.
- A graphics display set to "Millions of Colors" (24-bit or greater) is best. Displays set to "Thousands of Colors" (16-bit) do not have enough grayscale shades to adequately represent grayscale images.
- For some operations in IPLab for Macintosh, you also need the QuickTime™ extension to the operating system; however, this is not a requirement for most of the functions in IPLab. Apple's QuickTime™ can be obtained from Apple's website (http://www.apple.com).

You do not need any special boards or monitors in order to display and process high quality image data with IPLab for Macintosh. However, many cameras require special PCI boards (frame grabbers/ camera interface boards). Please consult the Extensions manual for the details on camera controls, frame grabbers, and cameras.

IPLab for Macintosh also allows you to create custom functions, called extensions, which you can integrate into the main application. In order to develop your own extensions you will need the appropriate development software. Consult the Writing Extensions chapter in this manual (starting on page 303) for the details on how to create your own extensions.
1.2  Setting Up *IPLab*

Before you start, we recommend that you make a backup copy of all of the CDs and disks in your *IPLab* package.

**NOTE:** *IPLab for Macintosh* is not copy protected. You may make as many backup copies as you wish *for use on only one computer at a time*. Please read the Software License Agreement for complete details.

The standard *IPLab* package contains a single CD-ROM holding all of the software you purchased (e.g. *IPLab*, camera controls, and extensions such as *Motion Control* and *MultiProbe*). To install your software:

1) Open the CD and run the *IPLab* installer, "Install IPLab 3.6," Follow the instructions given in the installer program. The installer creates a folder called “IPLab 3.6 Folder,” which contains the *IPLab* program.

2) Run the installers for the camera controls and for every extension you purchased. (E.g. the files "Install MultiProbe," "Install Camera Control …")

   When you select a destination folder for the installer, make certain you select the folder “IPLab 3.6 Folder.” That will install the extension or camera control into the right place. Extensions must be in the same folder as *IPLab*, and not inside a subfolder.

3) Some camera hardware comes with special drivers that we do not install. You will need to install these drivers, following the instructions that came with them.

4) If you installed any new camera controls or drivers, you must restart your computer.

We recommend keeping only one copy of *IPLab* on a computer. Doing so avoids confusion between the copy of *IPLab* you wish to use and the copy of *IPLab* that actually has your extensions installed.

New camera controls and updates to existing camera controls are posted on Scanalytics' World Wide Web site (http://www.scanalytics.com). You may also call Scanalytics to receive more information about camera controls (703-208-2230).

If you already have extensions for use with a previous version of *IPLab for Macintosh*, you should update them to the new version 3.6 extensions whenever possible. However, you may be able to transfer your old extensions to the *IPLab* 3.6 folder and have them work, if necessary. Be aware that some old *IPLab* extensions have to be updated in order to work with *IPLab 3.6*. 
1.3 Allocating More Memory to IPLab

Your computer divides its pool of RAM among multiple programs that can run simultaneously. You can tell the Mac operating system how much memory you want each program to take out of your total available RAM. To make certain IPLab has enough memory for operating on large amounts of image data, IPLab's installer sets IPLab to use 60% of your computer's available RAM. If your needs increase and you find yourself unable to open images, then you may want to increase the amount of RAM used by IPLab.

To allocate more memory to IPLab:

1. Make certain IPLab is not running. The icon should not be grayed-out.
2. Highlight the IPLab icon.
3. From the File menu, select Get Info and then Memory.
4. Increase the Preferred Size number and close the window.

Although IPLab does work with the virtual memory scheme built into MacOS, that method is so slow that we generally do not find it useful. Therefore, you should try to get by with allocating no more memory to IPLab than you have available in your total pool of RAM (less what is required by the operating system).

You are now ready to use IPLab.
Chapter 2 describes the most important parts of the IPLab environment. Becoming familiar with the features numbered in the image below will help you start using IPLab quickly and smoothly. Each numbered feature is described in a numbered part of this chapter (for example, the image window, feature #2, is described in the section, "#2: The Image Window."

Chapter 4, the Should-Read Section, describes some other useful features of IPLab. You may want to return to read Chapter 4 after you have started using IPLab. Chapter 5, the Operation chapter, details the inner workings of IPLab.
2.1 #1: The Menu Bar

*IPLab’s* commands are organized under descriptive menu names. The menus are titled **File, Edit, View, Enhance, Analyze, Math, 3D, Control, Ext(ensions), Script, and Window**.

The Menu Reference chapter of this manual is organized according to menu. So, for example, if you want information about a command in the **Enhance** menu, go to the section of the Menu Reference chapter that describes the commands in that menu. If you want to perform a measurement, for example, but don’t know what command to use, look in the section of the manual about the **Analyze** menu. As you would expect, all measurement and analysis commands are grouped there. The **Extension** menu items are discussed in the Processing Extensions chapter of the Extensions manual.

To find the command you need, look for the menu that describes the basic operation you want to perform:

- Archive images and analysis results in **Files** and recover them.
- **Edit** images: copy and paste, and modify the ROI, sequences, variables, function keys, and more.
- **View** images and/or data in various ways.
- **Enhance** images to make them look better.
- **Analyze** the image data to extract information about the image as a whole or about parts of the image.
- Perform **Math** upon measurement results or image data.
- Process and visualize **3D** data.
- **Control** cameras and microscope hardware.
- Use **Extensions** that expand the functionality of the basic program.
- Record sequences of commands in **Scripts** to be repeated later with ease.
- Readily move, duplicate, and organize **Windows** on the screen.

The menus in **IPLab** are organized around these different kinds of operations.
This is the complete list of IP Lab commands:

<table>
<thead>
<tr>
<th>File Menu</th>
<th>Edit Menu</th>
<th>View Menu</th>
<th>Enhance Menu</th>
<th>Analyze Menu</th>
<th>Math Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Image (F6)</td>
<td>Undo (Z)</td>
<td>Show/Hide Image</td>
<td>Normalization</td>
<td>Segmentation</td>
<td>Convert to 8/24 Bit (F7)</td>
</tr>
<tr>
<td>Open (F3)</td>
<td>Cut (X)</td>
<td>Home Image (H)</td>
<td>White Balance</td>
<td>Segment at ROI (F2)</td>
<td>Unsigned-&gt;Signed (F4)</td>
</tr>
<tr>
<td>Open As</td>
<td>Copy (C)</td>
<td>Select Frame (F4)</td>
<td>Equalize Contrast</td>
<td>Modify Segments</td>
<td>Change Data Type (F5)</td>
</tr>
<tr>
<td>Close Image (F6W)</td>
<td>Paste (V)</td>
<td>Show/Hide ROI</td>
<td>Image Ratios</td>
<td>Change Segment Color (F6)</td>
<td>Change Color Type (F8)</td>
</tr>
<tr>
<td>Save (F5S)</td>
<td>Clear</td>
<td>Show/Hide Segment</td>
<td>Invert</td>
<td>Set XY Units</td>
<td>Subsample</td>
</tr>
<tr>
<td>Save As</td>
<td>Crop Image</td>
<td>Show/Hide Drawings</td>
<td>Pseudocolor (F8E)</td>
<td>Define XY Units</td>
<td>Interleave Zeros</td>
</tr>
<tr>
<td>Revert to Saved</td>
<td>Set Value (F6E)</td>
<td>Show/Hide Labels</td>
<td>Edit Color Table</td>
<td>Set Measurements</td>
<td>Sort Data</td>
</tr>
<tr>
<td>Delete File</td>
<td>Set Color</td>
<td>Photo</td>
<td>Apply Color Table</td>
<td>Measurement Options</td>
<td>Barrel Shift</td>
</tr>
<tr>
<td>Foreign Formats</td>
<td>Select All (F6A)</td>
<td>Animate</td>
<td>Color to Grayscale</td>
<td>Measure ROI (F6R)</td>
<td>Swap 0 and 255</td>
</tr>
<tr>
<td>Export</td>
<td>Define ROI</td>
<td>View Options</td>
<td>Despeckle</td>
<td>Measure Segments</td>
<td>Binarize</td>
</tr>
<tr>
<td>View to File</td>
<td>Modify ROI</td>
<td>Image</td>
<td>Linear Filter</td>
<td>Quantify Segments</td>
<td>Pixel Arithmetic</td>
</tr>
<tr>
<td>View to Clipboard</td>
<td>Delete Frames</td>
<td>Text</td>
<td>Median Filter</td>
<td>Histogram</td>
<td>Image Arithmetic</td>
</tr>
<tr>
<td>Sequence to Movie</td>
<td>Transfer Frames</td>
<td>QGraph</td>
<td>Edge Filter</td>
<td>Row/Col Plot</td>
<td>Complex</td>
</tr>
<tr>
<td>Edit File List</td>
<td>Reverse Sequence</td>
<td>Perspective</td>
<td>FFT Filter</td>
<td>Contours</td>
<td>Set Pattern</td>
</tr>
<tr>
<td>Select File List</td>
<td>Change Drawings</td>
<td></td>
<td>Morphology</td>
<td>Transforms</td>
<td>Encode-Decode</td>
</tr>
<tr>
<td>Set Indexed Name</td>
<td>Clear All Drawings</td>
<td>View As</td>
<td>Define Linear Filter</td>
<td>Calibrate Intensities</td>
<td>Split Color Channels (F6L)</td>
</tr>
<tr>
<td>for Open</td>
<td>Select Next Drawing</td>
<td>Image</td>
<td>Flip</td>
<td>Dispose Segment Layer</td>
<td>Merge Color Channels (F6M)</td>
</tr>
<tr>
<td>for Save</td>
<td>Transfer Attributes</td>
<td>Text</td>
<td>Register</td>
<td>Extract</td>
<td>Convert Color Channels</td>
</tr>
<tr>
<td>for Delete</td>
<td>Set Row/Column Label</td>
<td>QGraph</td>
<td>Rotate &amp; Scale</td>
<td>Color Table</td>
<td>Affine Color Transform</td>
</tr>
<tr>
<td>Edit Image Info</td>
<td>Stamp Overlays on Image</td>
<td>Perspective</td>
<td>Mosaic</td>
<td>Rect. ROI to 1D</td>
<td></td>
</tr>
<tr>
<td>Define Image Info</td>
<td>Assign F Keys</td>
<td></td>
<td></td>
<td>Linear ROI to 1D</td>
<td></td>
</tr>
<tr>
<td>Edit FITS Header</td>
<td>Preferences</td>
<td></td>
<td></td>
<td>ROI Boundary</td>
<td></td>
</tr>
<tr>
<td>Page Setup</td>
<td></td>
<td></td>
<td></td>
<td>Image Sizes</td>
<td></td>
</tr>
<tr>
<td>Print</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quit (F3Q)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3D Menu</td>
<td>Camera Menu</td>
<td>Control Menu</td>
<td>Ext. Menu</td>
<td>Script Menu</td>
<td>Window Menu</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
<td>----------------------------------------------</td>
<td>------------------------------------------------</td>
<td>------------------------------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>3D Equalize</td>
<td>Camera Select &amp; Setup</td>
<td>Device Setup</td>
<td>&lt;Commands Added by Processing Extensions:&gt;</td>
<td>New Script</td>
<td>Show/Hide Tools</td>
</tr>
<tr>
<td>3D Filter</td>
<td>Focus</td>
<td>Device Select</td>
<td></td>
<td>Open Script</td>
<td>Show/Hide FKeys</td>
</tr>
<tr>
<td>3D ROI Sum Plot</td>
<td>Single Exposure</td>
<td>Device Specific</td>
<td></td>
<td>Run Script (G)</td>
<td>Show/Hide Status (U)</td>
</tr>
<tr>
<td>3D Mosaic</td>
<td>Full Acquire</td>
<td>&lt;Device specific commands&gt;</td>
<td></td>
<td>Set Variable</td>
<td>Show/Hide Variables</td>
</tr>
<tr>
<td>3D Oblique View</td>
<td>Acquire 3 Channel Color</td>
<td>Microscope Control</td>
<td></td>
<td>Continue</td>
<td>Dispose Window (K)</td>
</tr>
<tr>
<td>3D Stacked View</td>
<td>Acquire Bias</td>
<td>Microscope Variable</td>
<td></td>
<td>Cancel All</td>
<td>Dispose All Windows</td>
</tr>
<tr>
<td>3D Tracer</td>
<td>Acquire Dark</td>
<td>Microscope Speed</td>
<td></td>
<td></td>
<td>Rename Window (R)</td>
</tr>
<tr>
<td>3D Projector</td>
<td>Set Color Balance Values</td>
<td>Stage Control</td>
<td></td>
<td></td>
<td>Duplicate Window (D)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stage Set Zero Position</td>
<td></td>
<td></td>
<td>Move Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stage Record Position</td>
<td></td>
<td></td>
<td>Change Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stage Move to Recorded Position</td>
<td></td>
<td></td>
<td>Frames to Sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stage Z-Sections Setup</td>
<td></td>
<td></td>
<td>Sequence to Frames</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stage Speed</td>
<td></td>
<td></td>
<td>Tile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StageScan Array Calibrate</td>
<td></td>
<td></td>
<td>Stack</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StageScan Array Setup</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>StageScan Mosaic Calibrate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>StageScan Move/Iterator</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Must-Read Section: Quick-start with the IPLab Interface
2.2 #2: The Image Window

The image window contains the image, the ROI, and image overlays.

Image windows are also called data windows. Data of any kind can be displayed in an image window. The active window is the one with the highlighted title bar and which appears to be in front of other images. It is also referred to as the ‘front window.’

The Image Window, with Pointers to Parts of the Image

2.2.1 #2a: The Image

Your data is the image. In the example above, the image includes the fluorescent label as well as the black background.

As far as the IPLab program is concerned, the image is an *array of numbers*. An image is an orderly arrangement of pixels, and each pixel has a numeric intensity value. Hence, an image is an array of numbers. The easiest way to see this is to alternately view the image as an image and as text by using the View As command (View menu) to view the image as Text and then as an Image again.

IPLab/Mac images can contain three kinds of arrays: 1-dimensional, 2-dimensional, and 3-dimensional data. Any array of data, even a 1D array, is called an “image.”

- 1D data is a single row or column of numbers. 1D data typically is the result of some analysis, but you can also create a 1D data set.
- 2D arrays of numeric data are common images. 2D data may also result from some measurements or calculations of statistics or other information.
- 3D image data consists of a collection of 2D image frames. The third dimension can be either time (x,y,t) or the third spatial dimension (x,y,z). 3D images are also called image sequences.

You can only see one frame at a time in a 3D image sequence, so use the Select Frame command (View menu) to change frames. You can also press the Control and up- or down- arrow key to change frames.

Must-Read Section: Quick-start with the IPLab Interface
Examples of image windows:

![Example of 1D Data](image1.png)  ![Example of 2D or 3D Data](image2.png)

### 2.2.1.1 Image Coordinates

The individual elements of all images are referred to in this manual as *pixels*. The image's *coordinate system* pinpoints the location of each pixel. The x and y coordinates of any pixel are measured from the upper left-hand corner of the image. For multi-dimensional data, each image in the sequence is called a *frame*.

![Image Window Coordinate System](image3.png)

The coordinates of the top-left pixel in any image (or any frame of an image sequence) are (0,0). The first pixel to its right has coordinates (1,0); the first pixel below it has coordinates (0,1), *etc*. The bottom-right pixel has coordinates (Width-1, Height-1). This coordinate system is independent of how the data is zoomed or scrolled.

For multi-dimensional data, the first frame is frame number 0. Therefore, if the sequence has \( n \) frames, the last frame is frame number \( n-1 \). There are two ways that data can contain more than 2 dimensions:

- The data can be a time series of 2D images.
- The data can be a 3D spatial data set which consists of a single 2D image at multiple focal or time points.
#2b: The Region of Interest (ROI)

The Region of Interest is what it sounds like: the region or area of the image that you are most interested in. In practice, the ROI is the area of the image, any collection of connected pixels, that you have selected. The ROI is synonymous with the blinking, dashed selection line that surrounds the selected region. Other programs also call the ROI the selection box, the AOI (area of interest), the guard region, and the rubber band. The image only has one ROI at a time.

When viewing an image window as an image, the ROI is shown as a moving, dashed outline encircling the data elements (pixels) within the ROI. When viewing an image window as text, those data elements within the ROI are shown as red, underlined text. The Show/Hide ROI command (View menu) controls the visibility of the ROI's dashed line.

Rectangular ROI in 2D Data

ROI Showing in 1D Data

Freehand ROI in 2D Data

ROI Showing in Text View
2.2.2.1 Creating the ROI

You can use the ROI tools described on page 15 to select an ROI when viewing the data as an image.

If you are viewing the data as text, first click on a pixel to select it as a single-pixel ROI. Then scroll to another location in the window using the scroll bars. Now hold down the Shift key and click again.

You may also define regular ROI shapes via the Define ROI command in the Edit menu.

2.2.2.2 Uses of the ROI

You use the ROI for two reasons:

• To limit the area that will be processed or filtered.
• To define the area that will be measured or quantified.

Processing the ROI alone, instead of processing the entire image, allows you to apply different processing methods to different portions of an image. ROI processing is also much faster than processing the entire image. Many IPLab commands only affect the ROI, the selected region of the image.

2.2.2.3 Boundaries of the ROI

A rectangular ROI is specified by the left, right, top and bottom boundaries. Rectangular ROIs consists of all those pixels whose horizontal and vertical coordinates (h,v) satisfy:

- left ≤ h < right All horizontal locations between the left and right boundaries, including the left boundary but not the right boundary
- top ≤ v < bottom All vertical locations between the top and bottom boundaries, including the top boundary but not the bottom boundary

A rectangle, called the bounding rectangle, is associated with any non-rectangular ROI. It is the smallest rectangle enclosing all of the points in the ROI. The bounding rectangle is not directly visible in normal operation. The Status palette reports it as the bounds of the ROI.

For Advanced Users: Remember that the bottom and right bounds are not included within the selection. If, for example, the Status palette gives the right bound as 100 and the bottom bound as 50, then the pixel (100, 50) is just outside the bottom-right corner of the ROI.

Example: A Close-up of an Image

In this example, the left edge of the pixel at (100, 50) defines the right boundary of the ROI. The top edge of the pixel at (100, 50) defines the bottom boundary of the ROI.
2.2.2.4 Moving and Resizing the ROI

In a data window being viewed as an image or as text, you can move the ROI with the arrow keys on the keyboard.

You can move the ROI around a data window when it is viewed as an image by holding down the Option key.

- To move the ROI:
  1) Select one of the ROI tools from the Tools window.
  2) Hold down the Option key.
     Black handles will appear at the corners of the ROI when you hold down the Option key.
  3) Click in the center of the ROI, and
  4) Drag it to its destination.

- To change the size and shape of the ROI:
  1) Select one of the ROI tools from the Tools window.
  2) Hold down the Option key.
     Black handles will appear at the corners of the ROI when you hold down the Option key.
  3) Click down on one of the handles. The cursor will change to a diagonal crosshairs.
  4) Drag the handle to move that corner of the ROI.

You can also use the Modify ROI command (Edit menu) to move, contract, or expand the ROI by set amounts.

2.2.3 #2c & 2d: Segment and Drawing Overlays

Two layers sit on top of the IPLab image: the segment layer and the drawing layer. These two layers contain two different types of overlays that you can use to define measurement regions and to annotate images. In the diagram of an image window on page 9, item #2c is a drawing object annotating the image, and item #2d is a segment defining a region to be measured.

The segment and drawing overlays do not affect the data underneath them. You can draw graphics and text on top of your images without destroying the underlying data. You can think of these overlays as being like latex paint: you can paint the overlays on top of the data, and peel them off again without altering the data.

- The segment layer is like an image that lies on top of the real image data. Within its own layer, it colors each individual pixel one of eight different colors or transparent. It is used primarily for defining regions to be measured. Unlike the ROI, segments can define non-contiguous regions to be measured.
- The drawing layer contains drawings that you can select and move. It is primarily used for annotation. It is also like an image that lies on top of the real image data.

Overlays are only saved in the IPLab format. To see segment or drawing layers in TIFF-formatted images, use the Stamp Overlays on Image command (Edit menu). This command does overwrite image data in the regions of the overlay.
2.2.3.1 Creating and Editing Segments and Drawings

You draw into the two different overlay layers using the tools located in the middle of the **Tools** window. Pages 16 and 20 describe how to use the drawing and segment tools. Please also read "Tutorial 3: Particle Segmentation," on page 58, which covers this material.

You can also use menu commands to edit the overlays:

- **Segmentation, Segment at ROI, and Contours** will create segments. **Modify Segments** and **Change Segment Color** will edit existing segments. If the **Erase Segments…** option is chosen in **Measurement Options**, then **Measure Segments** and **Quantify Segments** will delete segments that do not meet criteria. **Dispose Segment Layer** deletes all segments.

The **Edit** menu contains commands that will edit, delete, and select drawings: the **Change Drawings**, **Clear All Drawings**, and **Select Next Drawing** commands.

In the **View** menu, you will find commands that let you hide and show the segment and drawing layers.

You can transfer the segment and/or drawing layers from one image to another using the **Transfer Attributes** command found in the **Edit** menu. You can use this feature to analyze the same regions in multiple images in a study: Define specific regions to analyze using segments in one image, and then transfer the segments definition to all other images in your study.

### 2.3 #3-6: The Tools Window

This section describes the **Tools** window, a.k.a. the **Tools** palette, which holds a large number of cursor-based tools. The first group of tool buttons (#3) defines regions of interest. The next bunch of tools (#4-5) can create and edit the segment or drawing layers. The last group of tools (#6), under the **Image** heading, help with scrolling, magnifying images, placing registration marks, rotating the color lookup table, and measuring lengths and angles.

![The Tools Palette](image)

**The Tools Palette**
#3: The ROI Tools

To create a region of interest, first click on the appropriate ROI tool in the Tools palette. Then use the mouse to draw the ROI.

You can define one ROI at a time. The ROI can have a variety of shapes, depending on the tool you use to draw it:

- **Rect. ROI:** Click this button to draw rectangular regions of interest. Draw the ROI by clicking down on the image and dragging the mouse. Hold down the Shift key while drawing to select square regions.

- **Oval ROI:** Use this tool to draw elliptical regions of interest. Draw the ROI by clicking down on the image and dragging the mouse. Hold down the Shift key while drawing to select circular regions.

- **Polygon ROI:** Use the polygon selection tool to select closed, polygonal regions. Click on each point of the polygon and double-click at the end to automatically close the polygon. You can also close the polygon by clicking on the starting point or by hitting any keyboard key besides the Escape key. You can cancel a partially drawn polygon by pressing Escape or Command-period.

- **Freehand ROI:** Click this button to freely draw closed, curved shapes. This tool gives you the greatest ability to follow an object's outline. To draw, click once and drag. The shape will be closed for you when you let go of the mouse button.

- **Line ROI:** Draw linear regions of interest using this tool, which is also called a multi-line or linear ROI tool. (This is like a closed polygon, except that the shape will not be closed after you double-click to end the drawing.)

  To finish drawing the line, double-click on the last point or press a keyboard key other than Escape. Press Escape or Command-period to cancel a partially drawn line.

  Use the multi-line tool only to create open curves or piecewise linear ROIs. You would use this tool with the Analyze menu command Extract Linear ROI to 1D, for example.

- **Wand:** The wand tool creates an ROI around a segment. Click the wand tool on top of or to the left of a segment. The ROI snaps to the outline of the nearest segment with the same color chosen in Segment part of the Tools palette (described on page 21). Clicking the wand on some extremely large, convoluted segments will select the rectangular area that contains the segment, rather than selecting the segment alone.

Notice that the ROI is drawn on top of the image. It does not affect the image data itself.

You may also define rectangular, oval, and linear ROI shapes via the Define ROI command in the Edit menu. This works on text-viewed images, as well.
2.4.1 ROIs in Text Windows

Within text-viewed windows, you can manually select only rectangular regions, because you cannot use the ROI tools on the text view. To select rectangular regions, click on one corner of the region. Then scroll across the image, hold down the Shift key, and click on the opposite corner. The selection will be red and underlined. You can also select an entire row or column by clicking on the appropriate heading in the left or top margin. You can select the entire image by clicking the white space in the upper-left corner of the text view.

2.4.2 Moving and Resizing ROIs

Within an image view, you may move the entire ROI: select an ROI tool and then hold down the Option key on the keyboard while left-clicking and dragging the mouse. You may also change the size of the ROI by holding down the Option key and selecting one of the ROI resizing handles with the mouse.

You cannot draw an ROI and paint a segment (with the segment tools, described below) or draw a drawing object (with drawing tools, described below) at the same time. After selecting a segment tool or a drawing tool, you will have to click on one of the ROI tools to draw an ROI again.

2.4.3 Scripting the ROI Tools

You can script the selection of an ROI tool. When recording a script, simply click on the ROI tool to record the tool's name. When the script runs that command, it will select the tool. You can then use the Alert command to prompt the user to select with that tool. For more detail, see the Scripting chapter on page 135.

If you want the script to select the ROI by itself, record the Define ROI command (Edit menu).

2.5 #4: The Drawing Tools

The Drawing Tools on the Tools Palette

The drawing tools add annotations, highlighting, and outlines to your image. You can also use drawings for other purposes, as well.* Drawings (a.k.a. drawing objects) exist within the drawing overlay, which is a layer independent of the image data. Drawings do not affect the image data.

* For example, you can define regions of interest with drawings. To do so, use the Define ROI command (Edit menu) to place an ROI at the selected drawing object.
The drawing tools occupy the middle of the Tools palette, under a label that reads either Segment or Drawing. Choose the drawing tools by clicking on the label Segment and selecting Use Drawing Tools:

![Segment](image)

Changing Between Segment and Drawing Tools

Select the drawing color by clicking on the color pop-up list below the Drawing label:

![Drawing Color](image)

Picking a Drawing Color Picking a Drawing Color within a Script

The segment colors are white, red, yellow, green, cyan, blue, magenta, and black.

Drawings may be outlined by dashed lines, indicating that they are selected. Left to right, top to bottom, the drawing tools are:

- Rectangular Drawing: Click this button to draw rectangular drawing objects. To draw, click once on the image and drag the cursor. Hold down the Shift key while you draw to produce a square.

  Double-click this button to open a dialog box that lets you set the attributes of the rectangle:

  ![Drawing Rectangle](image)

  **Attributes of Rectangular Drawings**

  When Fill is checked, the new drawings will be filled in with color, instead of hollow. Pen Size sets the thickness (in pixels) of the drawing's outline. Of course, you can only notice the line thickness when Fill is unchecked and the drawing is hollow.

  When you record this tool in a script, two new options appear. The Select Tool option activates the rectangle drawing tool for the script's user. You can then script the Alert command (Script Commands palette) to prompt the user to draw rectangles. The Custom option draws the rectangle when the user runs the script. In the Top, Left, Right, and Bottom fields, enter the rectangle's boundary positions as measured from the image's top-left corner.
Oval Drawing: Click this button to draw oval drawing objects. To draw, click once on the image and drag the cursor. Hold down the **Shift** key while you draw to produce a circle.

Double-click this button to set the **Fill** and **Pen Size** attributes of the oval, as shown for rectangular drawings on page 17.

Recording this tool in a script opens the same dialog that is shown for rectangular drawings on page 17’s right side. The script can either select the oval drawing tool for the user, or it can draw the oval drawing object itself.

Polygonal Drawing: Click this button to draw polygonal drawing objects. To draw, click once at each corner of the polygon. Polygons must have at least three sides. To close the polygon, click on the starting point or double-click anywhere within the image. You can also close the polygon by hitting any keyboard key besides the Escape key. You can cancel a partially drawn polygon by pressing Escape or Command-period.

Double-click this button to set the **Fill** and **Pen Size** attributes of the polygon, as shown for rectangular drawings on page 17’s left side.

Recording the polygon drawing tool in a script opens the same dialog box, allowing the script to select the tool and set the **Fill** and **Pen Size** attributes. Because polygons can have many corners, scripts cannot draw polygon drawings for you.

Line Drawing: Click this button to draw multi-faceted, linear drawings (also called multi-lines). Hold down the **Shift** key to draw lines along 90° and 45° angles. This is like a closed polygon, except that the shape will not be closed after you double-click to end the drawing.

To finish drawing the line, double-click on the last point or press a keyboard key other than Escape. Press Escape or Command-period to cancel a partially drawn line.

Double-click this button to set the attributes of the line:

- **Pen Size** sets the thickness of the line. Check **Arrow at End** to make your next line into an arrow.

When recording the line drawing tool in a script, two new options appear. The **Select Tool** option activates the line drawing tool for the script’s user. You can then script the **Alert** command (Script Commands palette) to prompt the user to draw lines. The **Custom** option draws the line when the user runs the script. Enter the line's end points, as measured from the image's top-left corner, in the **From** and **To** fields.
Freehand Drawing: Click this button to freely draw closed, curved shapes. To draw, click once and drag. The shape will be closed for you when you let go of the mouse button.

Double-click this button to set the Fill and Pen Size attributes of the shape, as shown for rectangular drawings on page 17's left side.

Recording the freehand drawing tool in a script opens the same dialog box, allowing the script to select the tool and set the Fill and Pen Size attributes. Scripts cannot draw freehand drawings for you.

Cut Drawings: Click this button to cut the currently selected drawing from the drawing layer. You can then paste it elsewhere, into any image's drawing layer, by using the button. Select the drawing with the tool; a dashed line will surround the selected drawing.

If you record the tool in a script, the script will cut the drawings that are selected at the time.

Annotation tool: Click this button to add text to the drawing layer. IPLab treats the block of text added by this tool as a single drawing object. This dialog lets you type and format your annotation:

The Font, Size, and Style settings are self-explanatory. Spacing can squeeze or stretch-out the text, either for compactness or legibility. Alignment controls the placement of the text with respect to its Position, which is given as (x,y) coordinates within the image.

When the Text radio button is chosen, a large text-entry field appears. You can type your annotation into that field. The date format used by the Date/Time option can be set using the Preferences command (Edit menu). When the Number radio button is chosen, fields appear for you to enter the digits and the number of decimal places with which to display it.

Scripting the text annotation tool opens the same dialog box. When this scripted command runs, it adds the text to the image's drawing layer.

Copy Drawing: Click this button to copy the currently selected drawing. You can then paste it elsewhere, in any image's drawing layer, by using the button. Select the drawing with the tool; a dashed line will surround the selected drawing.

If you record the tool in a script, the script will copy the drawings that are selected at the time.
Select Drawings: When this button is selected, you can click on top of drawings to select them. Hold down the Shift key to select multiple drawing objects. The draw, copy, and paste buttons act upon the currently selected drawing, as do the Define ROI (at Selected Drawing Object) and Change Drawings commands from the Edit menu.

You can record the Select tool in a script. When the script runs that command, it will select the Select tool for the script's user. You can then use the Alert command to prompt the user to select drawings.

Paste Drawings: Click this button to paste into the drawing layer. This will work with any drawings copied or cut using the Cut or Copy commands, or the draw, copy, or paste tools.

If you record the Paste tool in a script, the script will paste any drawings that have been copy or cut (that is, the contents of the drawing paste buffer).

So long as a drawing tool is selected, you can use the Cut, Copy, Paste, and Clear commands will work upon the currently selected drawing object.

2.5.1 Scripting the Drawing Tools

You can script each of the drawing tools. When recording a script, simply click (once) on the drawing tool to record the tool's name. The scripted behavior of each tool is described above, in the descriptions of each drawing tool, starting on page 16. For more detail on using these tools in a script, see the Scripting chapter on page 135.

You can also work with drawings in scripts by recording the following Edit menu commands: Change Drawings (to change the color of drawings), Clear All Drawings, and Select Next Drawing.

2.6 #5: The Segment Tools

IPLab's nine segment tools let you manually add segments like those created by the Segmentation command (Analyze menu). Segments define regions to be measured, so use these segments as the basis for measurements conducted by the Analyze menu commands (please see page 229). Segments exist within the segment overlay, which is a layer independent of the image data. Segments do not affect the image data.

The segment tools occupy the middle of the Tools palette, under a label that reads either Segment or Drawing. Choose the segment tools by clicking on the label Drawing and selecting Use Segment Tools:

Changing Between Drawing and Segment Tools
Select the segment color by clicking on the color pop-up list below the Segment label:

Picking a Segment Color

Picking a Segment Color within a Script

The segment colors are white, red, yellow, magenta, cyan, blue, transparent, green, and black.

You can set the pen size for any tool by double clicking on its button. The pen size is the width, in pixels, of the line painted by the segmentation tool (or of the line removed by the eraser). Also, double clicking on the first three segment tools lets you choose to paint a filled segment or not. Left to right, top to bottom, the segment tools are:

- Rectangular Segment: Click this button to draw rectangular segments. To draw, click once on the image and drag the cursor. To produce a square, hold down the Shift key while you draw.

  Double-click this button to open a dialog box that lets you set the attributes of the rectangle:

- Attributes of Rectangular Drawings

- Attributes when Scripted

When Fill is checked, the new segments will be filled in with color, instead of being hollow. Pen Size sets the thickness (in pixels) of the segment's outline. Of course, you can only notice the line thickness when Fill is unchecked and the segment is hollow.

When you record this tool in a script, two new options appear. The Select Tool option activates the rectangle segment tool for the script's user. You can then script the Alert command (Script Commands palette) to prompt the user to add rectangular segments. The Custom option adds the rectangular segment when the user runs the script. In the Top, Left, Right, and Bottom fields, enter the rectangle's boundary positions as measured from the image's top-left corner.
Oval Segment: Click this button to draw oval segments. To draw, click once on the image and drag the cursor. Hold down the Shift key while you draw to produce a circle.

Double-click this button to set the Fill and Pen Size attributes of the oval, as shown for rectangular segments on page 21's left side.

Recording this tool in a script opens the same dialog that is shown for rectangular segments on page 21's right side. The script can either select the oval segment tool for the user, or it can add the oval segment itself.

Polygon Segment: Click this button to draw polygonal segments. To draw, click once at each corner of the polygon. Polygons must have at least three sides. To close the polygon, click on the starting point or double-click anywhere within the image. You can also close the polygon by hitting any keyboard key besides the Escape key. You can cancel a half-drawn polygon by pressing Escape or Command-period.

Double-click this button to set the Fill and Pen Size attributes of the polygon, as shown for rectangular segments on page 21's right side.

Recording the polygon segment tool in a script opens the same dialog box, allowing the script to select the tool and set the Fill and Pen Size attributes. Scripts cannot add polygonal segments for you.

Line Segment: Click this button to draw multi-faceted, linear segments (also called multi-lines). This is like a closed polygon, except that the shape will not be closed after you double-click to complete the segment. Hold down the Shift key to draw lines along 90° and 45° angles.

To finish drawing the line, double-click on the last point or press a keyboard key other than Escape. Press Escape or Command-period to cancel a partially drawn line.

Double-click this button to set the attributes of the line:

- **Pen Size** sets the thickness of the linear segment.

When recording the line segment tool in a script, two new options appear. The Select Tool option activates the line segment tool for the script's user. You can then script the Alert command (Script Commands palette) to prompt the user to add line segments. The Custom option adds the line segment when the user runs the script. Enter the line's end points, as measured from the image's top-left corner, in the From and To fields.
**Segment Paintbrush:** Click the paintbrush button to paint freehand segments on the segment layer. To draw segments, simply click down on the image and drag the cursor.

Double-click the paintbrush button to choose the size and shape of the paintbrush tool:

![Segment Paintbrush dialog box]

**Attributes of the Segment Paintbrush**

Scripting the segment paintbrush tool opens the same dialog box. When this scripted command runs, it selects the segment paintbrush and sets the brush size and shape.

**Cut Segments:** Click this button to cut all segments within the ROI and stores them within a paste buffer. You can then use the [paste] button to place those segments within any image.

If you record the [cut] tool in a script, then it will cut segments when the script runs. It will cut all segments within the ROI, regardless of their color.

You can also use the **Dispose Segment Layer** (Analyze menu) to permanently remove all segments from the image.

**Segment Eraser:** The Eraser tool deletes the segment layer beneath the cursor. It deletes all segments, regardless of color.

Double-click this button to choose the size and shape of the eraser, just as for the Paintbrush tool.

Scripting the [eraser] tool opens the same dialog box to set the eraser's size and shape. When this scripted command runs, it selects the segment eraser and sets the size and shape.

**Copy Segment:** Click this button to copy all segments within the ROI. You can then use the [paste] button to place those segments within any image.

Scripting the [copy] tool copies segments when the script runs. It will copy all segments within the ROI, regardless of their color.

**Paste Segment:** Click this button to paste into the segment layer. This will work with any segments copied or cut using the **Cut** or **Copy** commands, or the [cut] or [copy] tools.

Scripting the [paste] tool pastes any copied or cut segments when the script runs.

So long as a drawing tool is selected, you can use the **Cut**, **Copy**, **Paste**, and **Clear** commands will work upon the currently selected drawing object.
2.6.1 Scripting the Segment Tools

You can script each of the segment tools. When recording a script, simply click (once) on the segment tool to record the tool's name. The scripted behavior of each tool is described above, in the descriptions of each segment tool, starting on page 20. For more detail on using these tools in a script, see the Scripting chapter on page 135.

You can also work with segments in scripts by recording many of the Analyze menu commands, including the Modify Segments and Change Segment Color commands.

2.7 #6: The Image Tools

The Image Tools on the Tools Palette

Scroll: When the Scroll tool is selected, the cursor appears as a hand. With this tool, you may scroll the data by clicking on and dragging in the image.

Rotate CLUT: When you click the Rotate CLUT tool on a 2D, indexed-color image, this tool rotates data through the range of colors used to display them. To be exact, it smoothly alters the image's color look-up table (CLUT) so that the same data is displayed with different colors.

For example, consider an image that has been pseudocolored so that the bright parts appear red and the dim parts appear blue. Holding the Rotate CLUT tool on the image will shift the colors: the bright parts will become blue, and the dim parts will become red. This does not change the data values of any pixels within the image, but it does change the CLUT. If you save the image after rotating the CLUT, the new CLUT is saved with it.

Press the Option key while the mouse button is down, and the CLUT will rotate in the opposite direction. While this tool is selected, you can use the up and down arrow keys to rotate the CLUT more slowly.

Magnifier Glass: Use the magnifier glass button to magnify or de-magnify the image.

Normally, the cursor looks like: . Clicking the mouse once doubles the image's magnification, making details appear larger and easier to see. The image window will expand to fit the image, if possible.

Hold down the Option key to see this de-magnification cursor: . While pressing the Option key, click the mouse once to demagnify (minify) the image, halving its magnification.

You can magnify up to eight times the original image size, and demagnify down to one eighth the original image size. The inside of the magnifying glass becomes blank ( ) when the image is at maximum or minimum magnification.
You can script the Magnify tool by opening a script and clicking on the magnifier icon in the toolbar. This dialog appears:

![Magnify Dialog]

The Select Tool option will set the current tool to the magnifier, without changing the magnification. The Change Magnification option will change the magnification of the front image. The Increase and Decrease options change the magnification by one step: 2X or 1/2X. The value in the Set to field can run from –3 to +3, corresponding to 1/8X, 1/4X, 1/2X, 1X, 2X, 4X, and 8X, respectively.

Registration Mark: This tool allows you to place up to ten registration marks in an image. The Register command (Enhance menu) uses these to align images. Click on landmarks within the image to place registration marks.

You can select registration marks by choosing the tool and clicking on the marks. Selected marks are highlighted in black. Hold down the Shift key to select multiple registration marks. As long as the Registration Mark tool is active, you can:

- Click and drag highlighted mark(s) to another location,
- Use the arrow keys to move highlighted mark(s),
- Delete highlighted mark(s) with the delete key.

With this tool active, the Select All command in the Edit menu selects all of the registration marks.

Length Measure: This tool lets you interactively measure lengths within your images. It measures length in the image's units, as set by the Set XY Units command (Analyze menu).

To use this tool, first click on the button. The cursor becomes a crosshairs and Length and Total (Total Length) fields appear near the bottom of the Status palette:

![Length Displayed on the Status Palette]

Next, click on the starting point of the object in the image. Then click on each successive point along the object being measured. The Length field displays the distance from the last click. The Total field displays the total length of the line from the first click to the current location. When you are done, double-click to end the measurement.

A text window named "ImageName.Lengths" will store the total lengths for each image. The measurement line will also become a numbered drawing on the image.
Hold down the Shift key to measure along a 45° or 90° angle.

Angle Measure: The Angle Measure tool lets you interactively measure angles (in degrees) within your images.

To use this tool, first click on the button. The cursor becomes a crosshairs and an Angle field appears near the bottom of the Status palette.

Next, trace an angle within your image by clicking on three points: the angle's side, its vertex, and the other side. The Angle field will display the angle until you click on the third point. When you click on the angle's final point, the measurement will be recorded to a text window named "ImageName.Angles." The line tracing the angle will also become a numbered drawing on the image.

Hold down the Shift key to force the angle's sides to a 45° or 90° angle with respect to the image.

2.7.1 Scripting the Image Tools

Except for the Magnify tool, you can script the selection of each image tool. When recording a script, simply click on the image tool to record the tool's name. When the script runs that command, it will select the tool. You can then use the Alert command to prompt the user to work with that tool. For more detail on using tools within a script, see the Scripting chapter on page 135.

As noted in its description above, the Magnify tool also lets you change the magnification of the current active image.

2.8 #7: The FKeys (Function Keys) Palette

The function key (FKey) palette's buttons represent the 12-15 function keys on the keyboard. You can assign IPLab commands to the function keys using the Assign F Keys command (Edit menu). Assign scripts to the F-keys by assigning the Run Script command (Script menu).

Instead of pressing the function key to perform the command or script, you can click on the FKeys palette instead. Clicking F1 would perform the same function as pressing the F1 key. This can be very convenient in a dark microscope room. The FKeys palette is also useful when you have activated the Mac OS's Function Keys feature, which can supersede IPLab's physical function keys.

You can name the F-keys by typing the names in the Assign F Keys dialog. You can narrow the palette to display only the F key numbers (F1-F15) by clicking the title bar's minimize box (the second box from the upper-right corner).
2.9 **#8: The Status Palette**

The **Status** palette displays information about an active data window. When it is visible, the **Status** palette’s title bar is always highlighted. It effectively appears to “float” over all other windows, but it is never considered the active window. To put away the **Status** palette, use the **Hide Status** command (**Window** menu) or click in its go-away box. Open the **Status** palette again with the **Show Status** command (**Window** menu).

- Coordinates of ROI in pixels and in the units of the active window
- Width and height of ROI in pixels and units
- Cursor coordinates in pixels and units.
- Data value at cursor
- Minimum, maximum image values
- Frame number
- Image width and height
- Image data type
- XY units of the active window
- Magnification level
- Length or angle
  (when using the tools: Measure Lengths or Measure Angles)
- Available memory

**Status Palette**
The **Status** palette shows the following information:

- Coordinates of the left, top, right and bottom sides of the ROI
- Width and height of the ROI
- Coordinates of the cursor's current location
- Value of the pixel under the cursor *
- Minimum and maximum pixel values in the image *
- Frame number and the range of frames in an image sequence (frames are numbered starting from zero)
- Width and height of the image
- Data type
- Units applied to the image by the Set XY Units command (Analyze menu)
- Magnification level
- Amount of memory empty and available for displaying images.

* This is a single value for indexed type (usually grayscale) images, or as three values corresponding to red, green and blue for direct type (color) images.

### #9: The Variables Window

The **Variables** window lists all of **IPLab**'s numeric variables. Variables are storage places for values used in commands. For example, **IPLab** stores the last-used exposure time in numeric variable #255. You can then use that variable as the exposure time within an acquire command (Camera menu). Variables are described in detail starting on page 119.

The **Variables** window is a special type of 1D data window. It provides you with 512 numbered variables of floating point data type that are always available. You cannot change the data type of the **Variables** window. However, in most other respects, these variables can be processed in the same way as any other data.

Variables are numbered from 0 to 511 (e.g. variable #0, variable #511). These identifying numbers are *indices* to the variables, and are listed as the row numbers to the left of the values.

You can enter any variable's value using the **Set Variable** command (Script menu), described on page 291. After selecting variables within the window, you can also use the **Set Value** command from the Edit menu (the keyboard shortcut is `z` or `z`). **Set Value** is described on page 187.

The **Variables** window is always available, but it may be hidden. Use the **Show/Hide Variables** command (Window menu) to open this window and to bring it to the front.
3  

Tutorials

*IPLab* has a lot of capability. You may need to tap only a portion of its power now, but after you get further involved in image processing and analysis, you may want to “go deeper.” This manual is designed to get you started as soon as possible. By working through the tutorials, you will get a feel for the power of *IPLab*, even if you do not need all its features right now.

It is best to do the tutorials in order. Some items in the later tutorials depend on techniques you would have learned in the earlier ones. However if you feel comfortable with Macintosh applications software, feel free to skip around. You may have to refer to the Menu Reference section of this manual to fill in details on some commands that were discussed in the tutorials you skipped.

3.1  

Tutorial 1: Getting Started

3.1.1  

Start *IPLab* and Open an Image

This is an easy tutorial just to get started. First you must start *IPLab*. There are actually several ways to do this. The surest method is the following:

- From the Finder, double-click on the *IPLab* program icon in the *IPLab 3.6* folder on your hard disk.

*IPLab* starts and you can begin working: either open an image or capture a new one or create a new image.

However, there is another way to start *IPLab*. This way is often used when you know that you want to open a specific image. First, quit *IPLab* by selecting the Quit command from the File menu, or typing Command-Q.

- From the Finder, find one of the sample images that came from the distribution disks. The image named “Washington” will do. Since that image has an *IPLab* icon on the desktop, the operating system knows it belongs to *IPLab*. You can double click on the image to start *IPLab*. This both starts *IPLab* and opens the “Washington” image.

You must be careful when using this feature of double-clicking on an image to open *IPLab*. If you have older versions of *IPLab* on your hard disk, you cannot be certain which version this process will start. The operating system starts the first version of *IPLab* that it finds, and this may not be the one you want.
Once *IPLab* is running, you will want to open other images. There is a menu command to do this:

1. Go to the **File** menu and choose the **Open** command, or type (Command-O).
2. Find and open the Images folder in your *IPLab 3.6* folder.
3. Select the Washington image by clicking once on its name. Since we saved this image with a “preview,” you may be able to see a preview of what the image will look like. If you have the QuickTime extension in your system Extensions folder, you can check the box called **Show Preview** and see the preview.

4. Since we are sure this is the image we want, click **OK** in the dialog.

The dialog disappears, a new window is created, and the Washington image appears within the window. The window is sized so that it shows all of the image data.

Another way to open an image is to find its icon on the desktop and drag that icon onto the *IPLab* icon (Leonardo da Vinci's portrait). If *IPLab* is already running, then this operation just opens the image. If *IPLab* is not already running, this operation first starts *IPLab*, then opens the image.
3.1.2 The Status palette

To see some information about the image, consider the Status palette. Glancing at the image window, you see a moving dotted rectangle around the entire image. That rectangle represents the Region of Interest, or ROI. The Status palette tells you that the ROI boundaries are L(Left) = 0, R(Right) = 256, T(Top) = 0, B(Bottom) = 256.

![Status palette](image)

Status palette

Also displayed are the width and height of the ROI (W and H).

As you move the cursor around within the image, several numbers are updated in the Status palette:

- The (x,y) coordinates of the cursor;
- The intensity value of the pixel at the cursor location (Val).

This position information is given in two columns, both of which say Pixels at the top. The first column always reports position coordinates in pixels while the second column is for reporting coordinates in calibrated distance units. You will learn how to create calibrated distance units later.

Below the intensity values, you see that the minimum and maximum values of the data are 0 and 237. Next, the line marked Frame shows that this is frame number 0, and that the minimum and maximum frame numbers are 0...0. If this were an image sequence, there would be more than one frame of image data in this window, and this line would read (0..n-1), where n is the number of frames. Next, you see that the image width and height are both 256 pixels, and that the image data is of type Byte, 8 bits per pixel, with potentially 256 gray levels.

The name of the calibration (“1 to 1 Pixels”) is displayed next to the Units label, and the magnification level is displayed below that.
3.1.3 Using the ROI and Image Tools

The Tools palette contains several icons which represent tools you use on images. They are in three groups with the headings ROI, Segment (or Drawing), and Image. First, consider the ROI tools at the top of the window. These tools allow you to draw a Region Of Interest (ROI) in one of several ways. ROIs will be discussed in more detail later, but right now you should practice with these tools.

1. If you do not already have the Washington image open, open it now with the Open command in the File menu.

2. Click on the rectangular ROI tool. This activates the rectangular ROI selection tool.

![Clicking on the Rectangular ROI Tool.](image)

3. Move the cursor into the image, hold down the mouse button, and drag it to create a rectangular ROI. When you let go of the mouse button, you see a moving dotted rectangle on top of the image to indicate the position of ROI.

4. Try the same thing with the oval ROI tool. When you let go of the mouse button, you see a moving dotted oval.

5. To draw a perfect square or circle using one of these two tools, hold down the shift key on the keyboard while you drag the cursor in the image.

![Drawing a rectangular ROI](image)

6. Now try drawing an ROI with the freehand tool at the bottom of this group of tools. Now when you click and drag in the image, the ROI exactly follows your cursor movements. When you let go of the mouse button, the shape is automatically closed with a straight line from the last point to the first one.

![Selecting the Freehand ROI Tool.](image)

The other ROI tools (except for the wand tool) work in a similar way, except in how you end drawing: To end drawing with the closed polygon or multi-line tools you must double-click at the last point you want. You can also end drawing a closed polygon by clicking once in the little flashing circle around the first point in the shape. You should experiment with these other ROI drawing tools.
In order to use the wand tool, you must have a segmented image, i.e. one in which you have applied the **Segmentation** command. We will wait until Tutorial 7 to show you how to use the wand tool.

You can also drag the ROI to a new position or resize it.

1. Draw a circular ROI using the oval ROI tool (Remember to hold down the shift key on the keyboard while you drag the cursor in the image to make the ROI a circle and not an ellipse).

2. Hold down the Option key on the keyboard. You will see four handles appear at the corners of the ROI. Move the cursor inside the ROI and see the cursor change to an arrow. In this state you can hold down the mouse button and drag the ROI to a new location.

3. Again hold down the Option key on the keyboard. When you will see four handles appear at the corners of the ROI, move the cursor over one of these handles. You will see the cursor change to an “x”. In this state, you can hold down the mouse button and resize the ROI.
Next, consider the tools at the bottom of the **Tools** window, under the heading **Image**.

1. Click on the magnifying glass icon at the bottom of the window. When you move the cursor into the image, the cursor changes to a magnifying glass with a plus sign in it.

   ![Selecting the Magnifying Glass Tool](image.png)

2. Move this cursor into the image and click once. The image and the window both grow in size.

3. In the **Tools** window, near the bottom, you see that the magnification now reads 2x. You can continue to click in the image until the magnification reads 8x. Each time you click the mouse, the image is magnified by a factor of 2, to a maximum of 8X. When you hold down the Option key on the keyboard, the plus sign in the magnifying glass changes to a minus sign. To de-magnify, click in the image while holding down the Option key.

4. To scroll the image, click on the hand tool in the **Tools** window, and then click within the image and drag it to the position in the window where you want it.

5. To rapidly move the image so that the top and left edges are aligned with the top and left edges of the window, you have several options:
   
   - Use the hand cursor to drag the image
   - Select the **Home** command from the **View** menu (Command-H)
   - Press the ‘Home” or “End” keys found on most keyboards
3.1.4 Using Segment and Drawing Tools

Return the image to magnification level 1. You can use the magnifying glass tool either with or without the Option key. Or, if you want to discard all changes made to the image, select the command Revert To Saved from the File menu. This closes the image and re-reads it from the last version saved to disk.

Now, consider the tools in the middle of the Tools window. The exact tools that appear there depend on which heading is showing: Segment or Drawing. You change which tools are showing by clicking in the pop-up list where the name appears.

Segment and Drawing Tools on the Tools Window

These tool sets let you draw into two different types of overlays. These overlays do not affect the image data at all. They can be seen or hidden, erased or stamped onto the image (stamping onto the image does affect the image data values).

The segment layer is like an image which lays on top of the real image data. It colors each individual pixel one of 8 different colors (or transparent). It is used primarily for defining regions to be measured.

The drawing layer contains drawing objects which you can select and move. It is used primarily for annotation.

Double click on each of these tools to see options about filling objects and the size of the drawing tool.

The first four tools are the same. They let you draw rectangles, ellipses, closed polygons, and multi-lines (i.e. piecewise linear objects). To draw a perfect square or circle, choose the rectangle or ellipse tool and hold down the Shift key while you draw in the image. If you hold down the Shift key while using the multi-line tool, the lines will be limited to 45° and 90° angles.

Segment tools include a paintbrush in the space where the drawing tools include a freehand shape tool. The paintbrush and freehand drawing tools are functionally similar in that you can create freehand drawings with either, but with segments it is more like painting into the overlay, and with the drawing tool it is more like drawing a boundary. Also, the freehand drawing tool always closes itself, but the segment paintbrush allows you to draw any shape.

There is an eraser among the segment tools. There is no eraser among the drawing tools because you remove drawing objects by selecting them and deleting or cutting them from the drawing layer. The drawing tools include a text drawing tool (which always brings up a dialog to prompt you for the text and its font characteristics), and an arrow tool which is used to select and move objects. Selected objects have a dotted outline.

Each set of tools has cut, copy and paste icons. The drawing cut and copy tools work on selected draw objects. The segment cut and copy tools work on that portion of the segment overlay which appears within the ROI. First, we will copy and paste selected drawing objects.
1. With the Washington image as the front window, select the drawing tools by clicking in the box where the word *Segment* or *Drawing* appears in the *Tools* window, and select *Use Drawing Tools* from the pop-up list.

2. Below that, click in the color box and choose the dark blue color.

3. Click on the freehand drawing tool. Then move the cursor in to the image and draw three or four random shapes.

4. Click on the arrow tool. Click within each of the objects you drew, one by one. You see that the one you click inside of has a dotted outline, which indicates that it is selected. You can select more than one object with the arrow tool by dragging a rectangle around several objects, or by holding down the Shift key while you click on each object. You can also de-select an object by holding down the Shift key and clicking inside it.

5. With several objects selected (their outlines appears dotted), click on the Cut Draw icon in the *Tools* window. The selected objects disappear. Now click on the Paste Draw icon. The objects are pasted back onto the overlay.

6. Click and drag inside any group of selected objects to drag them to another location within the image.

Now let’s work in the segment layer of the overlay.

7. Switch to the segment tools: click in the box where the word *Drawing* appears and select *Use Segment Tools* from the pop-up list.

8. Select any color you want, and then double click on the closed polygon tool. Check the *Fill* option.

   ![Segment Polygon Dialog](image)

   **Segment Polygon** Dialog. Get It by Double-Clicking On the Icon.

9. Next, draw several closed polygons on the image. You can finish each polygon by a double-click or by clicking in the little flashing ring around the polygon's first point.

10. Double-click on the eraser tool and choose a medium sized eraser, and then erase parts of the segment overlay. Notice that you cannot erase any of the drawing objects you created earlier.

11. Now, suppose you want to copy and paste a part of the segment overlay. First, select one of the ROI tools at the top of the *Tools* window. The freehand tool may be a good choice here. Then place an ROI around one of the polygons you drew.

12. Click on the Cut Seg. icon on the *Tools* window. The portion of the segment overlay which was inside the ROI is erased.

13. Now click on the Paste Seg. icon on the *Tools* window. That portion of the segment overlay is pasted back onto the image, but it has a flashing outline. This indicates that the paste is not complete yet. If you move your cursor inside that flashing outline, the cursor turns to an arrow, at which time you can click and drag the paste region. When you have dragged the paste region to the location you want, move the cursor outside the paste region and click again. This completes the paste.

14. Click again on the Paste Seg. icon. Again, you see the paste region with the dotted outline. This time you decide you don’t want to keep the paste. To cancel the paste, press the Delete key on the keyboard.

This tutorial is over. With this beginning, you are now ready to process image data. Quit *IPLab* by selecting the Quit command from the *File* menu. If any alerts ask you to save images, answer Don’t Save.
3.2 Tutorial 2: Image Enhancement

3.2.1 Overview of the Enhance Menu

Now let’s consider some of the image enhancement operations which are built into *IPLab*. Click on the Enhance menu and hold the mouse button to see what the menu contains. You will see that it is divided into four sections. At the top of the menu are commands that relate to contrast enhancement. Next, come a group of commands that relate to pseudocoloring. Following that is a group of commands related to image filtering. Finally, comes a group of commands related to geometric manipulation. You will get a chance to use most of these commands in these tutorials.

1. First start *IPLab* using one of the methods you learned in Tutorial 1.

2. Next open the Washington image using the Open command in the File menu.
3.2.2 Contrast Enhancement

Contrast enhancement is one of the most common applications of image processing. It is a simple way to help you see more detail in your images. First, we’ll talk about enhancing simple grayscale images, which have 256 gray levels (the most common kind). Then we will discuss grayscale images with more than 256 gray levels. Finally, we’ll discuss contrast enhancement on color images.

3.2.2.1 Adjusting Contrast on 8-Bit Grayscale Images

We will enhance the contrast in the image in a couple of ways. So the first thing to do is to make a copy of the Washington image and adjust the contrast on one of the copies:

1. From the Window menu, select the command Duplicate Window. Select the Active (Front) Window, Portion = Entire, and type in the new name “Manual Contrast Adjust”. Click OK, and a copy of the Washington window appears in a new window with the name Manual Adjust Contrast.

2. From the Enhance menu, select Normalization.

3. Click on the Expand/Collapse button (⊕) so you can see the histogram of the data within the ROI of the image (since the ROI is around the entire image, this is the histogram of the entire image).

The dialog shows a plot of the contrast function and the histogram of the data. The contrast function plot describes how values in the image are mapped into display intensities. Initially, the plot shows a diagonal line from the lower left corner of the plot to the upper right. This type of contrast function is the identity function. Display intensities are directly proportional to pixel values. There are several approaches to adjusting the contrast function plot. You should experiment with each of these methods:
a. Type values in for the minimum and maximum intercept points (at the bottom edges of the histogram).

b. Click and drag at the handles below the function plot.

One accepted procedure for enhancing the contrast of an image is to match the contrast function to the integrated histogram. Try this by first selecting Integrated from the Type menu. Then drag the sliders under the plot to approximately the locations shown in the figure below.

![Normalization Dialog Box](image)

Normalization Dialog Box

With this technique, you can make the linear contrast function match the histogram in a certain range of values.
However, you can apply a nonlinear contrast function that matches the histogram over a wider range.

1. Type in a **Mid Point** value of 85%.

2. Drag the endpoints in the plot, or type in values, so that the min. and max. are 0 and 184. The dialog should look approximately like the following figure.

![Normalization with a Nonlinear Contrast Function](image)

The gamma function is a nonlinear contrast function which, in this case, more closely matches the histogram curve. There are other uses for gamma contrast functions, for example to compensate for nonlinearities in the image capture process, or in the image display system. Of course, the real test of effectiveness is: “How good does the image look?” For this, we defer to your judgment.

Sometimes, you may want to enhance the contrast in only a portion of the image and you are willing to sacrifice contrast in other portions. We use the ROI to define the portion you are interested in.

1. **First, Cancel** this dialog, and then do the following steps to set the ROI to a portion of the image.

2. Click on the rectangle tool at the top left of the *Tools* window, directly under the *ROI* label. This activates the rectangular Region of Interest (ROI) selection tool.

![Selecting the Rectangular ROI Tool](image)
3. Move the cursor within the image and click-drag a rectangle similar to the following figure. You see a moving dotted rectangle on top of the image to indicate the ROI.

![Drawing a Rectangular ROI](image)

4. From the Enhance menu, select Normalization again. Click on the Use ROI Values button. The minimum and maximum parameters are updated.

5. Now adjust the contrast in any way you want to make the image look as good as possible. Click on the OK button in the Normalization dialog when you are finished.

**IPLab for Macintosh** also has an automatic method for adjusting the contrast so that the contrast function plot exactly matches the integrated histogram of the data within the ROI.

We’ll work on the original of Washington so you’ll later be able to compare results with your manual contrast adjustment techniques. The image Manual Adjust Contrast is currently the front window (it has its title bar highlighted).

1. Go to the Window menu and select the image named Washington from the list of images at the bottom of the menu. This brings the Washington image to the front. You can also click directly on the image.

2. From the Enhance menu, select Equalize Contrast. Check the Create New Window checkbox. Get the statistics from the entire image and change only the CLUT, which stands for Color Lookup Table. Click OK.

![Equalize Contrast Dialog Box](image)

**Equalize Contrast** Dialog Box

Another new window is created with the results of the contrast equalization command, which computed an “optimal” lookup table to apply to the image. This lookup table exactly matches the integrated histogram of the image data.

Now arrange the images on the screen so you can see all of them at once:

3. From the Window menu, select Tile.

You’ll see all of the images spread out on your monitor. You must decide whether the Equalize Contrast function gives a pleasing looking image, or if you prefer the results you got in Manual Adjust Contrast.
3.2.2.2 Adjusting Contrast on 16-Bit Grayscale Images

Now operate on a different image.

1. Go to the **File** menu and choose the **Open** command, or type (Command-O).

2. Find and open the 16Bit XRay image.

   The **Status** palette tells you that this image type is Short Int. Short Int images can have values between -32768 and +32767, and they require two bytes per pixel to store the image data (16 bits per pixel, instead of 8 bits). As you move the cursor around this image, you see in the **Status palette** very large pixel values. In fact the Min and Max values for this image are 335 and 3572 respectively. Since the monitor and display can only display at most 256 color or gray levels, **IPLab** must convert the image data with thousands of gray levels into a displayable range of values. This process is called normalization (in some fields it is called windowing and leveling). The default method of normalization is to map the minimum pixel value in the image to 0 and the maximum to 255 and all pixels with values between the min. and max. are linearly scaled to the 0-255 range. However, **IPLab** lets you change the normalization parameters to fix your attention on other ranges of pixel values.

3. Select the rectangle icon at the top left of the **Tools** window. This activates the Region of Interest (ROI) Selection tool.

4. Move the cursor into the image. Click and drag within the dark region of the ribs on the left side of the 16Bit XRay image.
5. From the **Enhance** menu, select the **Normalization** command.

6. Click on the Expand/Collapse button (△) so you can see the histogram of the data within the ROI of the image (since the ROI is around the entire image, this is the histogram of the entire image).

7. Grab the handle under the right edge of the plot and move it to the left. You’ll see the Max parameter change as you move the handle. Place the line so that the value in the Max field is approximately 1700. When you release the mouse button, the image should update to show different contrast. The apparent brightness and contrast of the image changes. This is because a smaller range of pixel values is mapped to 0-255.

8. There are several other options in this dialog for how to select the min. and max. values for the normalization mapping. Selecting the **Saturated Frame** option from the **Based On** pop-up computes min. and max. normalization parameters so that approximately 1% of the pixel values get mapped into 0 and 1% get mapped into 255.

9. Clicking the **Use ROI Values** checkbox causes the min. and max. parameters to change to saturate the image data within the ROI.

10. The options which refer to sequences apply only to image sequence windows, *i.e.* those which have multiple image frames stacked in a single window.

11. **Cancel** this dialog to reject all of the changes you have made.

Normalization does not destroy the original image data that has thousands of gray levels. It actually creates a copy of the image that is used only for display purposes. All data processing and analysis is done on the original data.

Use the **Dispose All Windows** command from the **Windows** menu to dispose of all of the open windows before proceeding.
3.2.2.3 Pseudocoloring

Another method of enhancing the contrast in an image is to pseudocolor it. A colorizing technique is to map each possible pixel value into some color, not just a gray level intensity. To do this, the pixel values are passed to a lookup table which has 256 entries. Each entry contains relative Red, Green, Blue (RGB) intensity values which are displayed at any pixel with value equal to that entry. To get gray intensities, the entries in the color table must have Red = Green = Blue. If this formula does not hold, then colors are displayed. For example, if entry number 100 has the RGB triplet (128,0,0) in it, then all pixels in the image with value 100 will display as a mid-intensity red, with absolutely no green or blue in them. Since the lookup table can generate colors, we call it a color lookup table, or CLUT. In this section, you will apply CLUTs to an image and edit them.

1. From the **File** menu, select **Open** and open the 3t3 Cell image.
2. From the **Enhance** menu select the command **Pseudocolor**.

![Pseudocolor Dialog Box](image)

3. There are several built-in color lookup tables. Choose the **Rainbow** option and click **OK**.

The image immediately changes from a grayscale image to a pseudocolor image. The pixel values are not changed, but the new CLUT has colors in all of its entries.

4. Go to the **Tools** window and click on the Rotate CLUT tool (the donut shaped icon at the bottom of the **Tools** window).

5. This tool allows you to rotate the CLUT interactively. Move the cursor inside the image and hold down the mouse button and you’ll see the colors in the image change as the CLUT rotates. To change the direction of rotation, hold down the Option key and the mouse button. For slower action, use the arrow keys on the keyboard to rotate the CLUT.

6. You can see what this CLUT looks like. At the bottom of the **Analyze** menu, there is a hierarchical menu called **Extract**. From the **Extract** menu, select **Color Table**. A new window is made which holds the Red, Green, and Blue entries in the CLUT. It is displayed as a graph, but you can also display the actual values by using “View as Text” in the **View** menu. You can use the scroll bars on the right side of the window to look at all of the entries. It is possible to edit this table directly, and re-attach the modified table to the image, but we’ll look at another way to edit the color table. To close this window which contains the CLUT, click in its go-away box in the upper left corner of the window and answer **Don’t Save** to the alert that appears.
7. From the **Enhance** menu, select the **Edit Color Table** command. Click on the label called **Show List** in the lower left corner of the dialog. From the scrolling list, select the **Blue Green Red** CLUT. The image changes to reflect the new CLUT. You can start from this CLUT to create a new one. As an aid in editing the CLUT, click in the **Histogram** box.

![Edit Color Table Dialog Box](image)

8. Click on the **New…** button and name the new CLUT “Black Red Yellow.” This name gets added to the scrolling list and you can begin editing it. The color ribbon at the top of the dialog shows you how the CLUT assigns colors.

9. To edit the CLUT, you insert anchor points within the table that tell how you want to treat that point. Move the cursor inside either the color ribbon or the boxed area in the middle of the dialog. As you do, the index box changes to show you the index value at the point of your cursor. Move the cursor to the far left where the index value shows 0. Click there and see a vertical line appear at the anchor point 0. Also the **Original** radio button becomes active and the color box matches the color of the table at this point. This is the original color at this index in the CLUT.

10. Your other choices among the radio buttons here are to make it a **Single** (new color), a **Solid** color from this anchor point up to the next anchor point (or the end of the table), or a **Ramp** of colors from this anchor point to the next. Choose the **Ramp** radio button. You will see the flag on the anchor point change to a triangle that points to the right.

11. We are going to set the color at this anchor point to black. Double click on the flag of the anchor point, or click once in the color box. The color picker appears and you can set a color. The exact form of the color picker depends on the version of the operating system you have. Set the color to black, *i.e.* 0 intensity, if it is not already at that value.

12. Place another anchor point at approximately 100. Make it a ramp by clicking on the **Ramp** radio button. Also set it to a 50% lightness red color (click in the color box to bring up you system’s Color Picker).
13. Place a third anchor point at approximately 200. Use the radio buttons to make it solid, and use the Color Picker to set its color to an 80% yellow color. The final CLUT editor looks approximately like the figure below. It has a ramp of colors from black to medium red then to medium yellow.

![Final, Edited CLUT](image)

14. You can edit the CLUT further by dragging the anchor points around. For example, you might drag the yellow anchor point left and right to see how the brightest pixels of the image are enhanced. Dragging the red anchor point affects the range of mid-value pixels.

15. Save the CLUT you just made and okay this dialog. The changes are left on the image. You can close the image without saving it.

3.2.2.4 Adjusting Contrast on Color Images

There is a color image on the distribution disk for you to play with.

1. Open the image “Color Histology.”

Notice on the Status palette that this image is Color 24 data type. This type of image does not need a CLUT to display color. It has three primary color channels, red, green, and blue, all in the same image, each with 256 levels of intensity (each channel has 8 bits and there are three channels for a total of 24 bits, hence the name Color 24). This provides the possibility to have millions of different colors in a single image, when you count all of the combinations of the primary colors. This is clearly more colorful than pseudocolor images which can allow at most 256 different colors in the same image. If your monitor is set to display millions of colors you will see the image at its best. If your monitor is set to display only 256 colors you probably will see some color contouring.

As you move the cursor around the image, you see the value field of the Status palette updates three numbers. These are the intensity levels of the red, green, and blue components of the image. For example, at the (x,y) location (50,80) the value field on the Status palette reads 197,138,161. This shows that the intensity of the red channel is much higher than the green and the blue channels, hence the image looks red there.
2. Select the **Normalization** command from the **Enhance** menu.

Remember that the color image has three components, red, green, and blue. The **Normalization** dialog has a pop-up menu which tells which color plane you want to work on. The pop-up menu also has an item called **All the Same**, which allows you to work on the overall brightness and contrast of the image.

![The Normalization Dialog Box](image)

3. Select the **All the Same** item from the pop-up menu labeled **Channel**. Select **Integrated** from the pop-up menu labeled **Type**. This shows the integrated histogram of the luminance or overall brightness of the image.

4. Adjust the slope and position of the contrast map line by grabbing and dragging the handles under the plot. This changes the min. and max. parameters and affects the overall brightness of the image.

5. Alternatively, you can select **Red**, **Green** or **Blue** from the **Channel** pop-up menu and adjust the contrast of that color channel separately. Notice that when you select the select Red from the **Channel** pop-up menu, the image reverts to the original and you begin adjusting the red contrast from there. The effects are not cumulative between selections from the **Channel** pop-up menu.

Here is something to keep in mind as you adjust the contrast on color images. When you select, for example, **Red** from the **Channel** pop-up menu, and adjust the min. and max. parameter values, you are affecting the pixel values of the red channel only, but you see effects which seem to enhance all of the colors.

6. Select **Red** from the **Channel** pop-up menu, set the min. value to 100 and press the Tab key on the keyboard. The image appears to be more blue-green. This is because everywhere in the image where the red plane had values less than 100 got set to zero, and most other red values got reduced. Since the red contribution is reduced, the other colors (blue and green) appear more pronounced.

7. With **Red** still selected, set the min. and max. values to 0 and 100 respectively. This really emphasizes the red channel so you see blue and green almost disappear.

8. Cancel the **Normalization** dialog so none of your changes are kept.
You see in the original image that between the red cells there is open space which has a slightly off-white tint to it. This region should in fact be nearly white. Such tinting often occurs as a result of an uncalibrated camera. We can adjust the overall tint of the image so that the background pixels are closer to white.

9. Select the freehand ROI tool from the **Status palette** and draw and ROI within one of the background regions as shown in the figure below.

![Image](image.png)

Outlining a Region of Background for **White Balance**

10. Before processing the image, let’s see what a distribution of the pixel values looks like in that region. Go to the **Analyze** menu and select the **Histogram** command. Fill in the dialog as shown in the figure below and click on **OK**.

![Histogram Dialog Box](image.png)
11. A new plot window is created. There are three curves plotted which indicate the number of pixels in each color plane with each possible pixel value from the red, green and blue planes. Although the three plots are clustered together, they are not right on top of each other. This indicates that the region has a slight off-white tint to it. This plot window is just for information purposes. It helps to verify that what you think you see with your eyes actually does correspond to the real data. This may not always be the case, especially if the monitor has a tint to it, in which case even perfectly white regions may look tinted.

12. With the Color Histology image as the front window and the ROI selected as shown above, select the White Balance command from the Enhance menu. You will see the entire image in the White Balance dialog with the ROI flashing. Move the flashing region around and click on Apply to see how this affects the white balance on the entire image. OK the dialog to keep your changes.

13. Let’s verify that the ROI region actually is whiter. Again, go to the Analyze menu and select the Histogram command. Fill in the dialog as before and click on OK. The histogram window will be updated to match the new data. In the resulting plot window, you’ll see that the three curves almost perfectly overlap, indicating that the data in that region is whiter.

This completes the tutorials on contrast enhancement. Go to the Window menu and select the Dispose All Windows command before going to the next tutorial session.
3.2.3 Filtering

In this session we’ll use the Washington image again. Open that image now. Since the Washington image is rather noisy, let’s try a process that reduces the noise.

1. From the Enhance menu, select Median Filter.
2. Check the Create New Window checkbox.
3. Click in the gray area of the mask until the solid black outline surrounds a 3x1 area of gray blocks as shown in the figure below.
4. Click OK.

![Median Filter Dialog Box](image)

A new window called Washington.F is created and the processed image is displayed in it.

The median filter reduces the noise, but also seems to blur the image somewhat. So let’s sharpen the Washington.F image with a linear filter. A sharpening filter is already included in the Linear Filter command dialog.

![Linear Filter Dialog with Sharpen1 3x3 Selected](image)

5. From the Enhance menu, select Linear Filter.
6. Check the box labeled Put Results in New Window.

Tutorials
7. Scroll the list of filters to the bottom, highlight the **Sharpen1 3x3** filter, and then click **OK**.

Another new window is created with the results of the sharpen filter. You may be interested in seeing the kernel which is used for **Sharpen1 3x3**, or you may want to create your own linear filter kernels. You can do both of these things using the **Define Linear Filter** command in the **Enhance** menu.

![Define Linear Filter Dialog Box](image)

Scroll down the list of built-in filters and click on the one called **Sharpen1 3x3**. The filter kernel for that filter is displayed so you can see it. You can experiment with other filters later. Right now, just click on the **Done** button.

There are several other ways to filter the image data. In the linear filter dialog you could have chosen the **Use Filter in Window** option. In this case you must have another data window on screen which contains the linear filter kernel you want to use. There are some filter kernels supplied on the distribution disks and you can create your own. For more information about that, see the section on **Linear Filter** in the Menu Reference chapter, on page 217. **IPLab** also lets you filter images in FFT space. For more information on doing that, see the section on the FFT Filter command. There is also an **Edge Filter** command with several built-in standard edge finding filters. See the description of the **Edge Filter** command on page 218 for more information on that.

### 3.2.4 Color Image Filtering

Here we consider two ways of filtering color images.

1. Open the image called “Color Image”.

2. So we can do a comparison of two different filter methods, duplicate the entire Color Image using the **Duplicate Window** command in the **Window** menu. Name the new image “Filtered Directly.”

3. With “Filtered Directly” as the front window, select the **Linear Filter** command from the **Enhance** menu. Find the filter named **Sharpen1 3x3** in the scrolling list of built-in filters. Highlight that filter, and do not check the box labeled **Put Results in New Window**. Click **OK**.

The image is filtered and looks like a sharpened version of the original. However there are some artifacts. For one thing, there appears to be a blue halo around the top of the apple. This is due to the fact that filtering this way applies the filter to the RGB components of the image. While this is often a fast and accepted way of filtering, this process can also introduce colors into the filtered image that were not in the original. Another method is to filter only the intensity information of a color image, not the color information.
Here is how you do it:

1. Bring Color Image to the front. Either click in its title bar or find it in the list of windows at the bottom of the Window menu.

2. Go to the Math menu and select the command Split Color Channels. Remember that a color image has three components, or coordinates: red, green, and blue. You could choose R-G-B and create three new image windows that contain these components. However, we are going to reduce the H-S-V coordinates (i.e. the blue halo). H and S stand for Hue and Saturation. They contain the color information in the image. The V coordinate is Value. It contains the intensity information. That is the one we will use for filtering. Click OK.

![Split Color Channels Dialog Box](image)

Split Color Coordinates Dialog Box

3. Three new windows are created to hold the H, S and V coordinate images. Go to the Window menu, find the one called “Color Image.V” (it should be the third one on the list at the bottom of the Window menu) and make it the front window.

4. Apply the Sharpen1 3x3 filter to this image using the Linear Filter command in the Enhance window. In this case put the results into a new window. When you click OK the new filtered image has the name “Color Image.V.F.”
5. Now that the intensity information is filtered, we must put this back together with the color information to create another color image. From the Math menu, select the command Merge Color Channels. The Merge Channels are HSV, and the Final Image should be Color 24. The names of the input windows are as shown in the figure below. Drag down each individual window to correspond to the appropriate color image. You are using the unchanged H and S coordinate windows and the filtered V window. When the dialog is filled out, click OK.

6. A new color image is created with a name like Untitled 1. We can change its name. From the Window menu, select the command, Rename Window. Enter the new name Filtered Intensity, and click OK.

7. Use the Tile command in the Window menu to spread out the images on the screen so you can see them. You may want to close the four intermediate images that were created, Color Image.V.F, Color Image.V, Color Image.H, and Color Image.S. You can click in their go-away boxes to close them. Don’t save them.

Seeing the images “Filtered Directly” and “Filtered Intensity” side by side, you will probably agree that the latter looks more like a real image. While it is handy and quick to use the Linear Filter command directly, you may want to consider this sequence of steps of color image sharpening for some applications. Don’t worry that there are too many steps to remember. IPLab has a way to encode a repetitive sequence of steps like this into a script which you can easily apply at the touch of a button. Scripting is covered in a later tutorial.
3.2.5 Geometric Operations

At the bottom of the Enhance menu is a group of commands for geometric processing.

<table>
<thead>
<tr>
<th>Flip...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register...</td>
</tr>
<tr>
<td>Rotate &amp; Scale...</td>
</tr>
<tr>
<td>Mosaic...</td>
</tr>
</tbody>
</table>

Geometric Processing Commands in the Enhance Menu

Use the Flip command to exactly rotate images by 90° increments or flip them horizontally or vertically. These operations are perfectly reversible.

Rotate and Scale can be used to perform arbitrary angle rotations and re-sizing. But, unlike the Flip command, Rotate and Scale is not reversible. When you rotate an image by 20°, for example, the coordinates of each pixel in the rotated image do not fall exactly on the coordinates of the original. So there must be some form of interpolation, which cannot be exactly reversed. So rotating by 20° then by -20° will result in an image that is slightly different than the original.

Use the Mosaic command to make one large image out of two smaller ones. You can join them vertically, or horizontally at the edges of the images. Or if there is an overlap between the images, you can place registration marks on each image to align them properly. Here is how you do that:

1. Open the files called “Slide DAPI 1” and “Slide DAPI 2.” These are two images taken from a microscope slide. Between Slide DAPI 1 and Slide DAPI 2 the stage was translated so that the field of view overlapped. Notice that where the two images overlap you can see at least part of a long chromosome in each. This is what we will use as a landmark to help register the images.

Slide DAPI 1 and Slide DAPI 2 Images: Overlapping Images of a Microscope Slide Showing the Placement of Registration Marks.

2. Make Slide DAPI 1 the front window (click in its title bar or select it from the window list at the bottom of the Window menu).
3. Pick up the “®” tool on the Tools window (click on the tool). Then locate and click on the centromere in the long chromosome (the bright part at the center). Place it exactly at cursor location (118,114). When you click in the image, you see a special symbol placed at the point that you clicked. This is the registration mark symbol. Do the same to place a registration mark at the corresponding point in the Slide DAPI 2 image. Place it exactly at cursor location (9,114).

5. From the bottom of the enhance menu, select the Mosaic command. Select the names of the two images, Slide DAPI 1 and Slide DAPI 2, by dragging the right arrow and select the Placement option Registration Marks. Click OK. A new image is formed which is the composite of the two. The second image (in this case, Slide DAPI 2) overwrote the first image in the region where they overlapped.

6. Keep this image on the screen. It will be used in the next session.

Finally, a word about the Register command. This command is used to align two images that are out of sync by pixel shift or rotation or scale. The primary way to use this command is first to place registration marks in the two images, similar to what you did in the Mosaic command example above. Then one of the images is considered the reference and the other image is the one which will be rotated, shifted and scaled so that its registration marks come into alignment with the reference image.

3.2.6 Adjusting the size of the image

Above, we gave you the exact coordinates for placing the registration marks. This resulted in a mosaic of exactly the right size. But in most cases the size of a mosaic based on registration marks will be different for each pair of images used. If you need to merge these images together (as in the next tutorial) or perform other operations, they will most likely all have to be the same size. There are two methods you can use to produce a fixed size image.

The first method is to use the Crop Image command, in the Edit menu, to trim the edges of the image. The Crop command keeps everything within the ROI and throws away the rest. If the ROI is not rectangular, the smallest rectangle that includes the ROI is used. The Define ROI command can be used to repeatedly set the ROI to a specific location.

The second method is to copy the existing data and paste it into a new image of the right size. It is assumed here that you have another window (the “model” window) of the desired size already open.

1. Make the Slide DAPI 1.M image the front window. Click on the rectangular ROI tool in the Tools window and select the command Select All from the Edit menu. This ensures that the ROI is around the entire image.

2. Next, select Copy from the Edit menu.

3. Make the “model” window the front window. Click on the rectangular ROI tool in the Tools window and select the command Select All from the Edit menu.
4. Go to the **File** menu and select **New Image**. Fill in the dialog as shown in the figure below and click **OK**. This creates a new blank image window the same size as the “model” image.

[![New Image Dialog](image.png)](image.png)

**New Image** Dialog to Create a New Window the Same Size as the Front Image

5. Now go to the **Edit** menu and select **Paste**. This pastes the image you copied from the original DAPI window into this New DAPI window.

6. You can drag the image around in the new window to position it. Once you click outside of the paste region the paste is completed, and the data is now in the new window.

### 3.2.7 Color Overlaying or Color Merging

Many people use a grayscale camera together with colored filters to capture high-resolution images at different source wavelengths. **IPLab** can be used to merge up to three of these grayscale images into a single high-resolution color image. You should still have the window called Slide DAPI 1.M (which was created in the previous session). This image was captured with colored filters matched to the DAPI probe. It will be merged with another image taken through a color filter matched to the FITC probe. Of course, there are many other sources of multi-spectral image data including remote sensing and laser optics, but this example will show you the basics. The command we will use here allows you to merge up to three images to form a color image. For merging images of more than three wavelengths, see the optional MultiProbe extension (described in the separate Extensions manual).

1. Open the image Slide FITC.

2. Go to the **Math** menu and select the command **Merge Color Channels**. The **Merge Channels** are assumed to come from **RGB**, and the **Final Image** should be **Color 24**. For the **G** (green) window, use Slide FITC, and for the **B** (blue) window, use Slide DAPI 1.M. There is no red channel, so clear the entry for the **R** (red) window. **OK** the dialog.
3. If you had made a mosaic from the images in the previous session exactly at the points we indicated, you should now have a new color image which is a color merge of the DAPI and FITC images. If instead, you got a message that said something about the images needing to be the same size, refer to the previous section for methods to change the size of an image.

3.2.8 Other Enhance Commands

There are two other commands in the Enhance menu, which we have not yet mentioned. They are Image Ratios and Invert, found near the top of the Enhance menu within the grouping of contrast enhancement commands. We’ll just mention how you might use them here, without actually applying them to any images.

The Invert command is used to invert the meaning of pixel values or CLUT values within an image. This is sometimes necessary when you get image data from another source. For example, some popular free imaging software packages encode images so that 0 = white and 255 = black. Of course this is backwards to standard scientific interpretation of intensities. Since for most functions you will prefer to have 0 = black and 255 = white, you need a way to convert those “foreign” images. The Invert command does this for you.

The Image Ratios command actually does several related things, depending on the option you select from the dialog. You can use it to:

• Take the ratio of two images in a very accurate way; or
• Use it to compute optical density; or
• Use it for “flat fielding” an image, i.e. correct for non-uniformities in the image capture process.

This completes the tutorials on basic image enhancement. You can quit IPLab by selecting the Quit command from the File menu, or type (Command-Q).
3.3 Tutorial 3: Particle Segmentation

3.3.1 What is a Segment?

When you want to quantitatively measure individual regions in an image, the first step is to separate, or segment, those particles from the background. Therefore in this manual we refer to a “segment” as any connected region of pixels which has been identified as being a single item of interest for measuring. The way segments are identified in IPLab is by the color of pixels in the segment overlay. The segment overlay is like an image that overlays the original image. In most places, you can see right through the segment overlay, that is, it is transparent. However, at pixels on identified segments, the segment overlay shows an opaque color.

You can have several different colored segments in an image. For example, you may be interested in two or three different kinds of particles in an image. In that case you would identify each kind of particle by a different color of segment overlay pixels.

There are several ways to perform segmentation. We will consider three here:

1) Manually; that is, drawing by hand;
2) Thresholding intensity values with possible manual touch-up;
3) Pre-processing for certain kinds of texture and then thresholding.

After any of the segmentation methods, you may want to apply some “cleaning” techniques (binary morphology) to improve the quality of the segments. We will also show you some of those techniques.

3.3.2 Hand Drawing

Manually tracing particles is a good place to start when understanding segments. In fact, there may be some cases where the image you are working with may be too difficult for any of the other segmentation techniques.

1. Find and open the image Three Phase from the Images folder. This is a metal alloy with three distinct phases represented by different intensities.

2. On the Tools window are several tools, which are used to draw into the segment overlay. Make sure that the word Segment appears in the Tools window as shown in the figure. If the word that appears in this place says Drawing then click on the arrow next to the word and select Use Segment Tools.

3. Below the word Segment in the Tools window is a color bar with an arrow next to it. If you click on that bar, you see a pop-up list of color choices. The original default color is the green near the bottom of the menu. Select this
color if it is not already selected. Now all of the drawing tools below that pop-up will use that color when you draw into the segment overlay.

4. The paintbrush tool, for example, is used to define freehand drawing segments. Double-click on the paintbrush. The dialog that appears allows you to select a paintbrush size and shape. Choose the third or fourth one, which are about the same size, except one is square and the other is round.

5. Now move the cursor into the image and use it to paint the inside boundary of two or three of the smaller white particles. You do not need to fill in the interior of the particle. Hole filling can be done later. However, you must be careful to create a fully closed region, especially if you eventually want to fill the interior holes or include the interior holes in the measurements.

It is important to note that you are painting into a non-destructive overlay on top of the image data. The segment overlay layer is used to define regions (or particles) which you want measure. You can remove the segment overlay, modify it, or just hide it, all without affecting your image data.

- You can draw as many segments as you want, using different colors for different particle types. To change the color, with which you draw the segments, simply click on the pop-up list at the color bar beneath the word Segment in the Tools window. The color bar changes and shows what color will be used for subsequent segment drawing.

- If you make a mistake, you can erase it with the eraser tool, which is also found on the Tools window below the paintbrush. As with the paintbrush, you can double-click on the eraser tool to change its size.

- There is one important thing to keep in mind about segments, especially if you want to do measurements on them later. Segments are defined as “connected sets of pixels.” Therefore, you cannot have two segments of the same color touching each other. If segments touch, they are really one set of connected pixels.
At any time you want, you can remove the segment overlay:

6. From the **Analyze** menu, select the command **Dispose Segment Layer**, found near the bottom of the menu.

After you have drawn all of the segments you want, you can measure all of the segments of a single color at one time. In the next session we show you additional methods of placing segments. If you would like to go directly to measuring segments, you can skip ahead to the next tutorial.

Now we’ll try another way of entering segments. You can close the Three Phase image, we’ll use a different one for the next session.

### 3.3.3 Placing a Segment at the ROI

Sometimes you may want to place many segments of exactly the same size and shape. In this exercise you will learn how to do that, using one of the ROI tools.

1. Find and open the image called “Gel.”

2. Since the image is small, let’s magnify it. Click on the magnifying glass tool on the **Tools** window, then click in the image once. You may want to “home” the image using the **Home** command from the **View** menu (Command-H) or the Home key on the keyboard.

3. Now select an ROI tool. Click on the rectangular ROI tool at the top of the **Tools** window.

4. Now draw a Region of Interest around one of the bands in the image.

5. There is a command that draws a segment at exactly the boundary of the ROI. From the **Analyze** menu, select the **Segment at ROI** command. Notice that it has command key combination Command-2. You may not see anything happen, but a segment was drawn. You’ll see it after the next step.
6. Now, instead of drawing a new ROI, let’s drag this one to a new location. Hold down the option key and move the cursor inside the ROI rectangle, you will see the cursor change to an arrow.

![With Option Key Down, Cursor Becomes Arrow Inside the ROI](image)

6a. With the option key still down, you can now click inside the ROI and drag it to surround the next band.

6b. Once you have dragged the ROI to the next location, you can see the segment that was formed under the ROI at the original location.

7. Now press Command-2 again, which does the command **Segment at ROI**.

8. You can continue this method of dragging the ROI and doing **Segment at ROI** several times. This creates a set of segments, all with the same size and shape.

**Note:** Remember that the segments must not be touching if you want to do measurements on them later.

You can also use any of the other ROI tools in this way to create multiple segments.

### 3.3.4 Copy and Paste a Segment

An alternative to the **Segment at ROI** command is to copy and paste segments. You have already learned how to copy and paste segments in Tutorial 1, but we can review the procedure here.

1. With the Gel image still as the front window, draw a single segment on the image. Don’t forget to first make sure that the **Segment/Drawing** name on the **Tools** window reads **Segment** so you have access to the segment tools.

2. To define what portion of the segments overlay to copy we need the ROI. So select one of the ROI tools at the top of the **Tools** window. The freehand tool may be a good choice here. Then place an ROI around the segment you drew.

3. Click on the **Copy Seg** icon on the **Tools** window. The portion of the segment overlay which was inside the ROI is copied.

4. Now click on the **Paste Seg** icon on the **Tools** window. That portion of the segment overlay is pasted back onto the image, but it has a flashing outline. This indicates that the paste is not complete yet. If you move your cursor inside that flashing outline, the cursor turns to an arrow, at which time you can click and drag around the paste region. When you have dragged the paste region to a location you want to keep it, move the cursor outside the paste region and click again. This fixes the segment in place.

5. Click again on the **Paste Seg** icon. Again you see the paste region with the dotted outline. This time you decide you don’t want to keep the paste. To cancel the paste, press the Delete key on the keyboard.

By copying and pasting you can create multiple segments of the same size and shape. What’s more, you can copy and paste multiple segments at once. Use one of the ROI tools to create an ROI around several segments:

6. Click on the **Copy Seg** icon on the **Tools** window. Again that portion of the segment overlay which was inside the ROI is copied, although this time it contains multiple segments.

7. Now when you click on the **Paste Seg** icon in the **Tools** window you see a larger portion of the segment overlay pasted, including multiple segments.
3.3.5  Intensity Thresholding

A common method of creating multiple segments at once is to use intensity thresholding. *IPLab*'s *Segmentation* command lets you determine the thresholds interactively.

1. Find and open the image Three Phase from the *Images* folder. This is an image of a metal alloy.

2. With the Three Phase image as the front window, you want to make sure that the ROI is around the entire image. If not, click on the rectangular ROI icon in the *Tools* window. This activates ROI drawing. Next, go to the *Edit* menu and select the command *Select All*. This places the ROI around the entire image.

3. Now you are ready to segment the image. Go to the *Analyze* menu and select the *Segmentation* command. Simply move the cursor on top of the Three Phase image and click inside any of the white particles (which is what you want to segment). When you click, a portion of the particles in the image will be automatically segmented. Now, click in another unsegmented portion of the particles. Keep doing this until all the particles are segmented. It should take about three or four clicks to segment the image.

![Segmentation Dialog Box](image)

4. Now try clicking on the (undo) button in the *Segmentation* dialog box. This will “undo” the last segmentation and revert those pixels outside the segmentation bin. You can click on the undo button multiple times and ultimately get back to the entire image unsegmented.
5. As you click inside the particles in the Three Phase image, you will see the min. and max. thresholds on the histogram being adjusted automatically. Each time you click in the image, more and more pixels are being included in the segmented region and this is indicated by the distance between the min. and max. thresholds increasing. Ultimately, you will see that the peak towards the right edge of the histogram is what is bound by the min. and max. thresholds. This peak represents the pixels inside the particles in the Three Phase image.

6. In the dialog, there is also a pop-up menu to let you define which color to use to define the segments. Choose the green color that is second from the bottom of the list.

7. Click OK. The dialog disappears and the image is painted in the segment overlay with a green color on the brightest phase particles.

Before you measure the segments, you may want to touch-up the global segmentation results.

There is a clump of three particles, which should have been separated from each other, but they are touching in the overlay. Since these segments touch, they would be measured as one object. Therefore, you must separate them before measuring if you want them measured separately.

![Portion of the Segmented Image Showing Touching Segments](image)

You can use the overlay drawing tools to correct this. Since the image is very small and you will be doing fine work, it is handy to magnify the view first.

8. Use the magnifying glass icon at the bottom of the Tools window and click once in the image near the clump of three particles. The image is now magnified to 2x. (As a reminder on how to use the magnifying glass tool, see Tutorial 1.) If you wish to scroll the clump of particles to the center of the window, click on the hand cursor, then click within the image and drag it to the position in the window that you want. Now that the view is magnified for easier use, you will use the eraser tool to correct the defects in thresholding.
9. On the **Tools** window are several tools which are used to draw into the segment overlay. Make sure that the word **Segment** appears in the **Tools** window as shown in the figure. If the word that appears is **Drawing**, then click on the arrow next to the word and select **Use Segment Tools**.

![Portion of the Tools Window Showing Segment Tools](image)

10. Among the tools available when you are using segment tools is an eraser (it is found beneath the paintbrush). Double-click on the eraser tool. In the dialog that appears, select one of the small-sized erasers.

11. Now move the cursor to the image and paint two lines to separate the clump of particles into a group of three. The eraser actually paints in transparent color into the segment overlay.

12. If you make a mistake, or you want to draw in an area which was not previously set by the **Segmentation** command, you can do that by hand drawing the segments as in the previous section of this tutorial.

13. When you are finished with this exercise, you can close the Three Phase image, since we will use another image for the next session. You can click in the go-away box of the window, or you can select **Close Image** from the **File** menu (Command-W), or **Dispose Window** from the **Window** menu (Command-K).

### 3.3.6 Cleaning Segments: Binary Morphology

In the last session you saw that you could clean up the results of the **Segmentation** command manually, using the tools on the **Tools** window to erase mistakes or draw in new parts of segments. In this exercise, you will use a computer-aided technique that is often called **Binary Morphology**. This is just a fancy name for a kind of filtering you can do to the overlay which defines segments. First, we’ll open another image to play with.

1. Open the image “Grains”.

2. Select the **Segmentation** command from the **Analyze** menu. This time, we’ll try a different method of segmenting the image. Set the lower threshold to about 189 by dragging the left handle under the histogram. Set the upper threshold to 255 by dragging the histogram's right handle. You can also type the values in the editable text boxes under the histogram's left and right edges. Choose the red color this time from the **Segment Color** pop-up menu near the top of the dialog box. Finally **OK** the dialog.
3. Since the image is rather small, you may want to use the magnifying glass tool from the **Tools** window to magnify the view of the image.

Notice that in general the segments are well defined over the grains. However, there are several segments with holes and there appear to be a few very small segments within the boundaries. These “errors” can be globally cleaned up using the **Modify Segments** command.
4. From the Analyze menu select the Modify Segments command. There are many options here to explore. First, select the Fill Holes option and Create New Window. Also, remember that you want to process the red colored segments so choose that color from the pop-up. Finally, OK the dialog.

5. A new processing image is created. You may have to magnify the image again using the magnifying glass tool from the Tools window. Only the overlay has been affected, but the holes within the segments on the grains are now filled in.

6. Select the Modify Segments command again. This time, we’ll use it to eliminate the small segments and clean up rough edges of all of the segments. Select the Open operation. Also, click on the squares within the Mask to make the shape rectangular as shown in the figure below. Click OK.

The Open operation performs an Erode and then a Dilate, using the mask you have selected. Erode reduces the size of every segment by eroding its boundary. Erode eliminates small segments. The Dilate operation then increases the size of the remaining segments to bring them back to approximately their original size.

While there are many other operations you can experiment with in the Modify Segments command, there is one more we’ll show you here.
7. Select the **Modify Segments** command again. This time, choose the **Boundary Only** option and **Create New Window**. Click **OK**. Only the boundaries of the grain segments are kept. If you want to return to the filled version, run **Modify Segments** again and choose the **Fill Holes** option.

You can close this image because we’ll use a different one for the next session.

### 3.3.7 Using Grain Boundaries

1. Open the image named “Multi-Grain.” This image has two different kinds of particles, which are distinguished by their size. We want to get a distribution of the sizes and then the mean intensity on the larger particles.

For this image, we’ll use a slightly different approach from the one we used in the previous sessions. We’ll actually work on getting the boundary pixels right, then take the complement of the boundary pixels to get the particles (taking the complement means to turn non-segment pixels into segment pixels while turning segment pixels into non-segment pixels).

2. Use the **Segmentation** command from the **Analyze** menu. Choose the green color for the segment color and set the max. threshold to about 100 with the min. threshold set to 0 by dragging the handles under the histogram. Then OK the dialog. This creates segments which cover the boundary pixels.

3. Now let’s work on this boundary to make it as thin as possible, but still able to separate the particles. Select the **Modify Segments** command and select the **Skeletonize** option. Be sure the segment color is the same as the one you used in the **Segmentation** command. OK the dialog. You will get an image which has thin boundary lines of green separating the particles.

4. The boundary lines are one pixel thick, which is actually too thin to adequately separate particles (the boundary line also breaks in some places). So select **Modify Segments** again and this time use the **Dilate** option with the **Mask** set as shown in the figure below. This thickens the lines and helps to connect them. It also does some minor damage to the boundary definitions of the smallest particles.

5. The last step is to take the complement of this segment overlay definition. Again, select the **Modify Segments** command. This time choose the **Complement** option and **OK** the dialog. The resulting image shows the particles in the segment overlay.

This multi-step procedure shows you that sometimes it takes a little ingenuity to derive a good segmentation technique. You can close all of the open images before proceeding.
3.3.8 Pre-Processing For Texture

The Segmentation command lets you set thresholds on the intensity values within an image. Sometimes particles cannot be separated from the background simply by setting intensity thresholds. In this session, you will learn one form of pre-processing on image data that will help separate particles based on a concept called texture. For a measure of texture, we use a simple technique of computing the difference between the minimum and maximum pixel values in a neighborhood around each pixel. While this is a very specific procedure for computing texture, the idea of pre-processing an image to enhance particles can be used in other scenarios as well.

1. Open the image “Curds&Whey.” Notice that this image has clearly identifiable particles of nearly uniform intensity on a background of highly variable intensity.

2. From the Analyze menu, select the Segmentation command. You can try adjusting the thresholds, but no matter what settings you use, the uniform particles cannot be segmented from the background because the background has some of the same intensity values within it. Cancel the Segmentation dialog to leave the image unchanged.

3. Now we will apply a series of steps that will result in an image in which the particles can be segmented by intensity. First, make sure that the ROI is around the entire image. If it is not, you must do two things: click on the rectangular ROI tool on the Tools window, then do the command Select All from the Edit menu.

4. From the Enhance menu, select the Morphology command. In that dialog select Erode and click inside the Mask to make the outline a full 5x5 pixels as shown in the figure below. Also select Create New image. This command works only within the ROI, which is why we had to use the Select All command first.

![Morphology Dialog for Grayscale Processing](image)

Morphology Dialog for Grayscale Processing

In the new image created by this command, each pixel is replaced by the minimum value of the pixels in a 5x5 neighborhood around that pixel.
5. Rename this image. From the Window menu, select the Rename Window command. Type in the new name “Min” and OK the dialog.

![Rename Window dialog for the results of the Erode operation](image)

6. Now bring the original Curds&Whey image to the front. Either click in its title bar or choose it from the list of windows at the bottom of the Window menu.

7. Again, make sure that the ROI is around the entire image.

8. And again select the Morphology command from the Enhance menu. This time choose Dilate and Create New Window, and then OK the dialog. Another new image is formed. To create this image, each pixel in the original image is replaced by the maximum value of all of the pixels in the surrounding 5x5 neighborhood.

9. Rename this image using the Rename Window command. Call it “Max.”

10. Now we have two windows which contain the min. and max. values in a neighborhood around each pixel. To take the difference between them, use the Image Arithmetic command in the Math menu. Fill out the dialog as shown in the figure below and OK the dialog. This takes the difference between the Max and Min images and places the resulting image into a new window called Texture.

![Image Arithmetic Dialog for Taking the Difference Between the Max and Min Images](image)
11. The image formed from this command shows brighter values where the texture in the original image is “rougher.” This image is amenable to thresholding. Select the **Segmentation** command. Adjust the upper threshold to some value between 110 and 130. This includes the particles and rejects the background fairly well. Finally, **OK** the dialog.

12. Now let’s transfer this overlay information back to the original image. From the **Edit** menu, select the command **Transfer Attributes**. You want to transfer just the segment layer attributes from the Texture image to the Curds&Whey image. When the dialog is filled in like the figure below, **OK** it.

![Transfer Attributes Dialog Box](image)

13. Now bring the Curds&Whey image to the front. Either click in its title bar or select it from the list of windows at the bottom of the **Window** menu. You can see the particles are well covered by the segments and ready to be measured. Or you can use the **Modify Segments** command as you did in a previous exercise to get the boundary only, or otherwise filter the segment definitions.

There were a lot of steps involved in finding the texture of the image. Don’t worry that you will not be able to remember all of the steps. That is why we made scripting, to record repetitive sequences of steps like that. More on scripting later.

### 3.3.9 Segmentation on Color Images

You can segment color images using the same **Segmentation** command under the **Analyze** menu. Since color images actually have three components to describe them (red, green, blue, or some transformation of these components) there are many ways to threshold color images. You must create a strategy for segmentation that includes components you want to examine and how to combine thresholds on the various components. The **Segmentation** dialog gives you a way of exploring those options.

1. Open the image called Color Histology (Color Histology image courtesy Dr. Lynn Heinel, Thomas Jefferson University). Notice on the **Status palette** that this is a Color 24 image. It is a stained histology section. To see it best, set your monitor display depth to millions of colors (use the Monitors control panel of the computer’s operating system). We are interested in the blue stained nuclei.

2. From the **Analyze** menu, select the command **Segmentation**.

3. Move the cursor over to the image and click on any of the nuclei in the image. They will all get automatically segmented pretty well. This is how powerful the **Segmentation** command in **IPLab** is. See what effect changing each of the **Sensitivity** slider bars have on the segmentation by undo-ing the previous segmentation (by clicking on the Undo button in the dialog), changing the slider position and then clicking inside the nuclei again.
4. If you wish to know what is going on behind the scenes, click on the **Histogram** tab. In the first pop up menu on the **Histogram** tab, select **Bs**. This is a color component which is equal to the blue value of each pixel divided by the pixel brightness. It gives a good definition of the blue color. Since we will not need any other color components to help us define the regions we are interested in, you should select **Off** in the pop-up menu in the next row.
5. To define the Min and Max thresholds on Bs, drag the left threshold so that the min. value reads approximately 90, and the right threshold all the way to the right edge of the histogram. When you let go of the mouse button, you will see segments defined in the overlay on the image as green (or the color you have selected as the Segment Color).

**OK** the dialog and the image is covered with an overlay which defines the segments.

This was an easy example. Now let’s try a more difficult one that will require thresholding on more than one color component.

1. Open the image called Color Image. This is also a Color 24 image. The apple has a bright spot on it where incident light is reflected. It is sort of pink. Suppose this is the color we are interested in.

2. Use the Freehand ROI to outline the bright spot on the apple. This will limit the histogram in the segmentation dialog to just these pixels.

3. From the **Analyze** menu, select the command **Segmentation**.
4. Select the yellow segment color. It will contrast with the other colors in the image, making it easier to see which pixels have been segmented.

5. In the first pop up list on the Histogram tab, select $R_s$. Also select the Off option from the pop-up menu on the next row as shown in the dialog above.

6. Under the histogram of the $R_s$ values of the pixels in the ROI, drag the thresholds to approximately the values shown in the figure above. You will see pixels in the image appear red where the $R_s$ values are between these thresholds.

7. The selected pixels include not only the area we are interested in but also parts of the shadow and the leaf. This is because those regions also have some red in them. Here is where you must get creative. What other color characteristics distinguish the desired area from the undesired areas already selected? If you said the desired areas are brighter, you’re right, and there is a way to select only the bright areas. First change the Off to And. This lets you access another component to use. Select Y (which is a symbol for luminance or brightness) in the second component pop up list. This time set the thresholds to approximately the values shown in the figure below. You will see the green segment overlay pixels in the shadow areas disappear.

![Segmentation Dialog Box After Applying $R_s$ and $Y$ Thresholds](image)
8. You are almost done. There are still some pixels in the leaf which pass these two thresholds. So turn the second Off to And and select Hue. **Hue** corresponds roughly to wavelength, or color in a generic sense. Notice that the histogram, which appears, is separated at the two ends of the plot. Also the color bar above the histogram indicates where the red hues lie, at the outside ends of the plot window. Click on the Invert button ( ) and adjust the thresholds to approximately the values shown in the figure below.

---

Segmentation with \( R_s \), Y and Hue Thresholds

9. This is about the best we can do, so click **OK** to finish up.
10. There are still some stray pixels in the segment overlay. You can use the Modify Segments command to clean up segments on color images just like on grayscale images. First, use Select All from the Edit menu to place the ROI around the entire image. Then select Modify Segments from the Analyze menu and apply the Open filter with a large kernel. This will remove small groups of segment pixels while leaving the large group over the bright spot intact.

![Modify Segments Dialog Box](image)

3.3.10 Summary

We have spent a lot of time on segmentation. But that step is often the most difficult and critical step in measuring particles. There are basically two methods of segmenting an image: drawing the segments by hand and creating segments by thresholding (with the Segmentation command). We also showed you one pre-processing option which you can use to consider texture as the criterion for segmentation. Once an image is segmented, you can still clean up the segments, again either by using the segment drawing tools again or by using the Modify Segments command.

The next step in the analysis process usually is to measure the segments. That is the topic of the next tutorial.
3.4 Tutorial 4: Particle Measurements

Often the objective of electronic imaging is to measure things that you cannot conveniently measure any other way. This may include densities (i.e. intensities), areas, and shape information, even the location of particles. You may also want summary information such as the percentage of the total area that is occupied by certain kinds of particles. There are several commands in IPLab which help you set up and perform those kinds of measurements. In this tutorial you will learn how to use them.

We will use the Multi-Grain image in this tutorial. Find and open it now. Now let’s look at the procedures you might use to set up the measurements: Set the distance units, define the parameters you want to measure, and specify any annotation you want on the image to help you recognize the measured segments. Keep in mind that not all of the procedures are required every time you do a measurement. Usually you will have to do these setup steps only once, then you can do many measurements, all with a single command.

3.4.1 Setting Distance Units

Often the first step in measuring anything is to set up the distance units. Let’s assume that we know this image was captured with a specific 10x microscope objective and that we know the size of each pixel in microns. We can set up IPLab to take this information into account whenever we deal with image captured by this objective lens.

1. From the Analyze menu select the command Define XY Units. This command lets you build a list of possible “units” from which you can choose. At the start, there is only one defined unit on the list, “•1:1 Pixels.” There are several ways to enter additional defined units. In this case we will use some knowledge which we assume we already have about the size of the pixels in an image when we use this objective.

2. In the dialog click on the New… button. Another dialog asks you for the name you wish to attach to this units definition. Type in “My 10x” and click OK. Back in the Define XY Units dialog enter the formula “1 pixels = 10 µ(10x).” (To enter the “µ” character, hold down the Option key and type “m”.) The dialog should now look like the figure below. Click on the Save button. If you have other known units, you could define them in a similar way right now. For now, just click on the Done button and the dialog disappears.
3. Now that distance has been set up, you can attach that units definition to the Multi-Grain image. From the Analyze menu, select the command Set XY Units. In the pop-up list you will see My 10x which you entered in the previous step. Select that option. Notice that you could also place a calibration bar in the image. This would go into the overlay of the image so it would not overwrite any of your image data. But for now, simply leave this option unchecked, and OK the dialog.

4. Now in the Status palette, you can see certain location coordinates given in µ(10x).

All size and position measurements are now done in the units of µ(10x).

IMPORTANT: If you define additional units in the Define XY Units dialog, you should try to keep the text you use for the units definition distinct. So, for example, if you decide to keep this units definition, you should not use the term “µ(10x)” for any other units definition.

Now that you have set the units on this image to be µ(10x), that information is saved with the image (if you save it as an IPLab image file, more on that later). Also, any new image that you create using the New Image command from the File menu, or any new image that you capture using one of the commands from the Control menu will have this distance unit definition.

What if you want to transfer this distance unit definition to another image, or set of images? The Transfer Attributes command of the Edit menu does this. The User's Guide describes the command on page 194.
3.4.2 Setting Up Measurements

There are two commands to help you set up the types of measurements you want to perform. Both are found in the Analyze menu: Set Measurements and Measurement Options. These commands are usually performed once before a set of measurements is done. In fact you do not even need to have an image open in order to use these two commands.

1. First, select the command Set Measurements. This dialog lists many measurements you can perform on each particle. Check the boxes labeled Sum, Mean, Area, and Minor Axis. These parameters give basic information about each particle that is measured. The Sum parameter is the sum of all of the pixel values within the particle, while the Mean is the sum divided by the number of pixels in the particle. The Area will be measured in the distance units you have attached to the image. The Minor Axis is a measure to the diameter of each particle (it is in fact the width of an ellipse that is fitted to the shape of the particle). It is also measured in the units you have attached to the image. We want this parameter so we can limit the size of particles we will accept.

2. To limit the size of particles we’ll accept, check the box next to Minor Axis under the column titled Limit?. Enter 100 for the Min, 100,000 for the Max. This will reject all of the small segments, those less than 100 µ(10x) (which corresponds to 10 pixels, recall from our defining formula of 1 pixel = 10 µ(10x)). Since we expect no particles over 100,000 µ(10x) (or 1000 pixels) in diameter, this will measure only the big ones.

3. You don’t have to do anything about the checkbox that reads Single Measure: Column-ordered, or the line below it, Max. Number of Segments, since you are performing more than one measurement.

4. It is sometimes handy to create a map of the segments, classified by color according to their relative sizes (or some other criterion). To do this, check the box labeled Classify According To, and from the pop up list select Area.

5. Finally, click on the OK button.

6. Next, select the command Measurement Options from the Analyze menu. We must decide which color segments we will be creating and measuring. Select yellow from the Segment Color pop-up box. Fill in the other check boxes as you see in the figure below. We will include interior holes in the segments as part of the measurements since we don’t expect the grains to actually have holes. Later on, we’ll show an application where we don’t want to include the interior holes.
In most cases you should ignore segments that touch the edge of the ROI. When a segment touches the edges of the ROI, you cannot be sure how big it really is, so its area computation biases the distribution of areas of whole particles. We will choose to annotate the image by drawing an outline in the drawing overlay layer around each particle that is measured. Choose the red color from the pop-up list of color for the **Outline/Number Color**. You can enter any name you want for the results window. The default of 1000 measurements will be adequate for this tutorial. If you plan to measure many particles over multiple images, you may want to increase this number. Since we are measuring particles in a single image, we won’t label the rows in the results window. But if you were measuring particles in multiple images, you might want to place the image name on each row of the results window. Finally, **OK** the dialog.

![Measurement Options Dialog Box](image)

**Measurement Options** Dialog Box
3.4.3 Measuring Segments

You must first segment the image in order to measure its particles. While we used a rather elaborate technique to segment the Multi-Grain image in the last tutorial, we’ll just use a simple one here.

1. With the Multi-Grain image as the front window, select the **Segmentation** command from the **Analyze** menu. Enter 110 and 255 for the Min and Max values, choose the yellow segment color, then **OK** the dialog.

2. This gets most of the segments, but leaves many of them touching. To separate them, go to the **Analyze** menu and select **Modify Segments**. Fill in the dialog as in the figure below. Be sure you use the yellow segment color. **OK** the dialog. An image called Multi-Grain.F is formed with better separated segments.

![Modify Segments Dialog to Help Separate Touching Grains](image)

3. Once setup is done, the measuring step is trivial. Now measure all of the segments in Multi-Grain.F at once by selecting the command **Measure Segments** from the **Analyze** menu.

The segments which satisfy the criteria are measured and outlined. A new Measurement Results window is created behind all of the existing windows. In addition, if you checked the box **Classify According To** in the **Set Measurements** dialog, there is also another image window behind the results window which shows the color coded classes of particles. If you have other measurements to perform, for example on other images, you could do them now, and the results will be collected in the same **Measurement Results** window (until you reach the maximum number of measurements). To view the results, bring the **Measurement Results** window to the front by selecting it from **Window** menu or by clicking in its title bar.

![Example Results Window](image)

Don’t discard this results window; we will use it in the next session to give a distribution of the mean intensities of the particles. This results window can be saved as a text file so you can import it into a spreadsheet or word processing document. You can also copy and paste the contents of the Measurement Results window to a spreadsheet or word processing document.
Let’s also look at the window called “Multi-Grain.Classes”. Select this window name from the bottom of the **Window** menu. You see this window has all of the measured particles separately color coded by their area. This image contains floating point data. If you move the mouse cursor into the image and watch the value field on the **Status palette**, you see that the pixel values on each particle are all of the same. In fact the pixel value within each particle is exactly the area value which is calculated by the **Measure Segments** command and placed into the table of results. You can close this image (click in its go-away box in the upper left corner of the window).

Classes Window Showing the Particles Color Coded by Size

Now we’ll repeat the measurement command, but this time just to measure a few of the segments. (In a later tutorial, you will learn how to measure a single particle by pointing to it with the wand tool.) First, let’s remove the red outlines that show each measured particle.

4. Make sure the Multi-Grain image is the front window (its title bar should be highlighted). If it is not, select it from the list of windows at the bottom of the **Window** menu. From the **Edit** menu, select the command **Clear All Drawings**. This removes all of the red outlines from the drawing layer overlay.

5. Now go to the **Tools** window and click on the freehand ROI tool below the polygon ROI tool.

Selecting the Freehand ROI Tool on the **Tools** Window
6. Move the cursor into the Multi-Grain.F image and draw an ROI around a few of the segments. Be sure to fully enclose all of the segments you want to measure, because any segments that touch the ROI boundary will be ignored.

![Placing an ROI Around a Few of the Segments](image)

7. Since we want these measurements kept separately, go to the **Measurement Options** command in the **Analyze** menu and change the name of the results window to be **Limited Results**.

![Changing the name of the results window in Measurement Options](image)

8. Now do **Measure Segments** again. This time, only the segments within the ROI are outlined, and the new results window contains measurement results for only those segments.

Now you can continue to outline groups of particles and measure them, or you can capture new images and segment them, then measure segments in those images. You do not need to return to the **Set Measurements** or **Measurement Options** commands again unless you want to change some of the those options.

There is also a wand tool that helps you point and measure individual segments. You will see later how to use that in a later tutorial.

Before continuing with the next lesson, close all of the windows except the one called Measurement Results.

### 3.4.4 Distribution of Parameter Values: QGraph

Now let’s use the measurement results to get a distribution of the mean intensity levels on the segments. You can use the same techniques to get the size distribution.

1. Find the Measurement Results window and make it the front one (select it from the list of windows at the bottom of the Window menu). Looking at the **Status palette**, you can see that the height of this window is probably around 60. Since there is one measurement per row, this number tells how many particles were measured.
2. Click at the top of the column labeled **Mean**. That action selects the one column as the ROI. The data values within an ROI are shown in red underline when the data is displayed as text. Notice that when you clicked at the top of the **Mean** column, only the values in that column get displayed in red underline.

3. Now get a histogram of the mean values: From the **Analyze** menu, select the **Histogram** command. Make the number of bins 20. Select **on ROI** and check the boxes labeled **Use Data Minimum** and **Use Data Maximum**. Then **OK** the dialog.

![Histogram Dialog Box](image)

4. A new plot window appears. This plot is called a QGraph (using the Bar option, making it a bar graph). We will now make this plot look pretty so you can use it directly in reports.

![Original Histogram Plot of the Mean Densities of the Measured Particles](image)

5. Click-and-hold on the label “Bin Value” at the bottom of the plot. A pop-up menu shows several options. Select **Label**. In the dialog that appears, type in “Mean Intensity” and click **OK**.

6. Click-and-hold on the Y-Axis label, which currently reads “Pixel Count”, and from the pop-up menu select **Label**. Type “Particle Count” for the new label and click **OK**.

7. Click-and-hold on the title line of the plot which currently reads “Histogram of Measurement Results”, and from the pop-up menu select **Label**. Type “Distribution of Mean Intensity” for the new label and click **OK**.
8. Again, click in the plot window and select **X Axis** from the pop-up menu. In the dialog that appears, uncheck the **Use Data Min./Max.** option and type in your own min. and max. values. We used 100 and 200 since those numbers spanned the min. and max. mean intensity values found in the original plot, although you can use any numbers you wish for the **Minimum** and **Maximum** scale values. Set the decimal places to 0.

![Changing X Axis Options in the Plot](image)

9. You can size the plot by clicking in the lower right corner of the plot window and dragging it.

![Example QGraph Plot of the Mean Intensity Distribution](image)

You can copy and paste the QGraph plot to another application, such as the Scrapbook or a word processor, you can print the QGraph plot, and you can export it as a PICT or TIFF file.

Dispose of all of the open windows before proceeding.
3.4.5 Measuring Donut Shaped Regions

Sometimes, you need to measure particles with holes, and you do not want to include the holes in the measurements.

1. Find and open the image called Gray Histology. This image has several dark circular objects whose area must be determined. (The Gray Histology image is courtesy Beth Ann Nobel, H.T., Thomas Jefferson University)

2. When working with PICT or TIFF files, you may look at the Status palette and notice that the units attached to this image are µ(10x). This distance unit definition was the last one you set. IPLab automatically assigned the last units you specified to PICT and TIFF files. You can change the units back to 1:1 Pixels using the command Set XY Units.

3. Select the Segmentation command from the Analyze menu. Set the Min to 23 and the Max to 95. Select the blue color and OK the dialog.

Segmentation Dialog for the Gray Histology Image
4. Now clean up the segmentation using **Modify Segments** twice. The first time use the **Open** option with the mask set to the largest (5x5). The second time, use the **Close** option also with the largest mask. Each time be sure you have the color set to blue. These two applications help to remove small particles and also smooth out the edges of the segments.

![Modify Segments Dialog for the Close Operation](image)

5. Now set up the measurements we want to do with the **Set Measurements** command. In this case we’re going to measure only a single parameter, area. Make sure all of the other boxes in the dialog are unchecked as shown in the dialog below.

6. For this case, we’ll also check the box labeled **Single Measure: Column Ordered**. This will report the results for the different segments in columns instead of rows. Since we do not expect there to be more than 10 segments, leaving the default value of **Max. Number of Segment** equal to 10 does not hurt anything. This defines the width of the results table, which you should choose to be a little bigger than the number of segments you expect to measure. You will see that the results table will contain some zero valued columns for segments which don’t exist, but this does not matter.
7. Next, select the **Measurement Options** command from the **Analyze** menu. In this case, **DO NOT** Include interior holes since it is precisely the donut shaped regions which are of interest. Also, let’s number the segments as well as outline them. Be sure that the **Segment Color** is blue and OK the dialog.
8. Finally, select **Measure Segments** from the **Analyze** menu. A new Measurement Results table is created in the background. Make it the front window. You see that the columns are labeled “Area#1,” “Area#2,” etc. These are the area values for each of the particles found in the image. Since there were likely less than 10 particles found in the image, some of the columns have zero in them.

![Measurement Results Table](image)

**Single Measurement, Column-Ordered Results**

If we want, we can get the area of the holes also. We perform the measurement again, this time including holes. Then we take the difference between the results.

9. Make the image Gray Histology the front window again.

10. Remove the outlines and numbers: From the **Edit** menu select the command **Clear All Drawings**.

11. From the **Analyze** menu, select **Measurement Options**. This time, check the box **Include Interior Holes In All Measurements**. Change the name of the results window to “Measurement Results Filled.” **OK** the dialog.

12. Do **Measure Segments** again. Now you have another results window, this one called Measurement Results Filled. You can see it by selecting it from the list of windows at the **Window** menu's bottom.

13. We want to take the difference between this results window and the results window from the previous measurement to get the area of the holes. From the **Math** menu select the command **Image Arithmetic**. You want to subtract the results including holes from the results not including holes. Fill in the dialog as in the figure below. Then **OK** the dialog.

![Image Arithmetic Dialog](image)

**Image Arithmetic** Dialog to Compute the Areas of the Holes
14. A new window is created which has the results of the difference for each column.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Areas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1505.99</td>
<td>10235.02</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

Computed Areas of Holes

Dispose of all of the open windows before going on to the next tutorial.

3.4.6 Measuring Intensity Changes Over Time

In the previous session, you saw how to create column ordered results when measuring a single parameter. One primary application of this is in measuring intensity changes over time. Say you identify several cells whose intensity you wish to monitor. If the cells are immobile, you can do the following:

1. Use the Set Measurements command to select only the sum or the mean intensity values to report. Also, check the box to place the results into a column ordered results table.
2. Capture a single image. Use the segment drawing tools to outline the cells of interest.
3. Periodically capture another image into the same window (the option reads Into Front Window in most acquire dialogs) and measure the segments.

When you are finished with the series of image captures and measurements, each column of the table gives you a time history of the intensities within each cell.

This repetitive series of steps is best done within a script. If you want to jump ahead to learn how to make scripts, see the tutorial on scripts later in the manual.

3.4.7 Area Fraction and Other Particle Summary Results

The Quantify Segments command measures area and intensity information on all of the segments and reports a single line summary. Quantify Segments uses the limits defined in Set Measurements and the additional conditions defined in Measurement Options when it determines whether to include a specific segment in the computation.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tot</td>
<td>96.00</td>
<td>12062.00</td>
<td>180.00</td>
<td>22.61</td>
<td>0.06e-6</td>
<td>102.07</td>
<td>189.00</td>
<td>232.00</td>
<td>192.17</td>
<td>6.12</td>
</tr>
</tbody>
</table>

Example Results Window from the Quantify Segments Command

You can use Quantify Segments to determine, for example, area fractions, i.e. the percentage of total area occupied by a specific type of material or stained specimen. IMPORTANT: For area fraction calculation accuracy you should probably NOT ignore segments which touch the edge of the ROI and you should probably NOT include interior holes in the measurements.
3.4.8 Manual Measuring

There are several more commands in the Analyze menu which help you to measure particles and regions.

The Measure ROI command performs all of the measurements which you specify in the Set Measurements command, but it does this on the region defined by the interior of the ROI. Use this as a quick way to get a measurement on a region, without drawing segments. Notice that this command has the command key Command-1.

You can also measure lengths of particles manually using the Measure Lengths tool. Select the tool from the Tools window, click on one end then double click on the other end of a particle. To measure the length of curved lines, click once at each point along the curved boundary and double-click when you are done. Each subsequent measurement is added to the results window.

In the same manner as measuring lengths, you can manually measure angles using the Measure Angles tool.

For both Measure Lengths and Measure Angles, numbered lines are drawn in the object drawing layer of the overlay to show where you have done the measurements. These can be deleted by using the Drawing arrow tool to select and then cut or delete the objects.

3.5 Tutorial 5: Extracting More Intensity Information

Besides particle analysis there are several additional commands in the Analyze menu which help you extract information from images. Many of these are designed to give you different ways of looking at the intensity levels within an image.

3.5.1 Extracting Plots of Intensities

The histogram command gives a count of the number of pixels at each possible gray level in the front image or data window. You have already used the Histogram command to get a distribution of the mean intensity within each particle. You can also use it to get the histogram of intensities within an image. Open the 16Bit XRay image and try the Histogram command in Analyze window. The result is reported in a QGraph plot which can be exported or copied and pasted into another application.
Example Histogram Plot with Pop-Up Menu Showing Selection of Bar vs. Line Plot

The histogram is reported as a bar graph. You can change this to a line graph and back as follows:

1. Click in the plot window.
2. In the pop-up menu that appears, select Gallery, and then from that menu, select Line or Bar. A line graph is shown above.

Next is the Row/Column Plot command.

1. To use this command, first open the Gel image.
2. Select the rectangular ROI tool from the Tools window and draw an ROI from the top of the image to the bottom, exactly covering one lane of the gel.

Make an ROI to Cover One Lane of the Gel
3. Now go to the Analyze menu and select the Row/Column Plot command. Choose the Row Sum on the ROI and OK the dialog. For each row of the ROI, this gives a plot of the sum of all pixel intensities.

![Row/Column Plot Dialog Box](image1)

Result of Row/Column Plot Applied to the ROI of the Gel Image.

The Dotted Outline Is the Rectangular ROI

4. The areas under each peak represent the total sum of all of the pixels within each band. You can have IPLab measure these areas. First, select the Set Measurements command in the Analyze window and set it up to measure only the sum (and don’t select column ordered results for this one). Now use the rectangular ROI tool and drag an ROI around one of the peaks. Then select Measure ROI from the Analyze menu (or type Command-1). Continue to draw new ROIs around each peak and measure all of the peaks. You can also drag the ROI to the left and right by holding the option key down and clicking within the ROI.

(For a more detailed gel analysis product, ask Scanalytics about ONE-Dscan.) The results of these measurements go into a Measurement Results window which is placed behind all of the other windows.

Sometimes, you’ll want to find the pixel values along some slice through the image. (NOTE: This is sometimes called a “histogram” although the term “histogram” is correctly applied only to the distribution of pixel values).
5. Open the 16Bit XRay image or make it the front window. Select the multi-line ROI tool (below the oval ROI tool in the Tools window). Draw a line along the spine of the subject. If necessary, you can click at more than one place to make a piecewise linear ROI. At the end double click to finish.

6. Go to the Analyze menu. At the bottom is a hierarchical menu called Extract. From that list choose Linear ROI to 1D. The result is presented in a basic plot window, giving the values of the pixels along the Linear ROI which you drew. Alternatively you can view this information as text or a QGraph plot (Select the view you want from the View As hierarchical menu at the bottom of the View menu).

7. Another alternative to get the values along a boundary is the Extract: ROI Boundary command. Bring the 16Bit XRay image to the front (click in its title bar or select it from the bottom of the Window menu). Again, go to the bottom of the Analyze menu and select the ROI Boundary option from the Extract submenu.

8. The results are presented as a three column table showing the x and y coordinates of each point along the ROI and the intensity values at each point. This command can be applied to any shaped ROI, not just piecewise linear ROIs.
3.5.2 Density Calibration

There are nonlinearities in most image acquisition systems. Some, such as film scanning, are known to be worse than others are. If you have a standard image with known intensities, you can correct for these nonlinearities. The idea is to create a table with measured intensity values and known intensity values then use this table as a mapping function which transforms each pixel value in an image into calibrated intensity values.

In this session we will again work with the image called 16Bit XRay. Open that image or bring it to the front if you have it already open.

1. There are two support files which we need to calibrate the intensities in this image. Find and open the files called Standard Image and Standard Values.

2. The Standard Image contains squares of known density values. We will treat this image as if you had captured it from a specimen with known density squares. We have already drawn in segments which will help us measure the mean values within each of the squares. Normally, you would do this yourself.

3. Go to the Analyze menu and select the Set Measurements command. Choose only the Mean value parameter (Do not check the Single Measure: Column Order Results option. You want these results row-ordered.). OK the dialog.

4. Select the Measurement Options command and make sure the Segment Color is green. It will not be necessary to outline or number these segments. Name the results window “Measured Intensities” and OK this dialog.

5. Finally, with the Standard Image as the front window, select Measure Segments. To see the results bring the Measured Intensities window to the front.
6. Column number 1 forms part of the table we need. The other part of the table is found in the Standard Values window. Now we are ready to calibrate the intensities on the 16Bit XRay image. From the Analyze menu, select Calibrate Intensities.

7. The image you want to change is the 16Bit XRay. The measurements are found in the window named Measured Intensities, in column number 1. The standard values are found in the window named Standard Values, in column number 0. You can experiment with the best fitting curve, but we have found that the Power curve works best for this set of data. This curve shows how the data will be transformed. When your dialog looks like the figure, OK it.

A new image is created. This window contains floating point data for the best possible accuracy, calibrated to values between 0 and 10. You can now save this image as a calibrated intensity image and do all your intensity measurements on this.
3.6 Tutorial 6: Scripting

3.6.1 Basic Scripting

Researchers need to create their own protocols, teachers and students need to experiment with alternative methods, and technicians often need to automate repetitive tasks. The method IPLab uses to address these issues is the script. A script is simply a sequence of commands selected from the menus.

In this tutorial you will learn how to make and edit scripts. Let’s start with an existing script which is distributed with IPLab.

1. From the Script menu, select the Open Script command.
2. Find the Tutorial Scripts folder within the Scripts folder within the IPLab 3.6 User Folder.
3. Click once on the script called Background Subtract, and then click on Open Script. A script window appears.

The commands that make up the script are shown in a scrollable list in the script window. The Background Subtract script performs a background subtraction on the Chromosomes image. To estimate the background, the script applies a series of morphological erosion filters (erosion computes the minimum value in a neighborhood). Then the script subtracts that estimated background from the original image.

(Before continuing with the script itself, it is worth mentioning that there is another common problem in the image capture process. This problem is due to “uneven illumination.” Although many people think of using background subtraction for this problem, there is a much better method to correct for uneven illumination, called “flat field correction.” This is covered in the command Image Ratios under the Enhance menu.)
Whenever a script window is open, a new floating palette appears, the **Script Commands** window:

![Script Commands window](image)

You will notice that some of the commands in this window (Cut, Copy, Save, Print, etc.) are duplicates of commands in the menus. But these commands apply only to the front-most script window, not to image windows. When a script is open the commands in the menus are recorded into the script, so you must use the buttons in the Script Commands window to operate on the script itself.

Scripts are opened, by default, with Recording turned off, so that they can’t be changed by accident. To examine or edit a script, recording must be on. The **Record ON/OFF** button at the top of the script controls this.

1. Click once on the **Record OFF** button, so that it changes to **Record ON**.

You can look at the dialog and parameters which are used by each command in the script.

2. Double-click on the first **Morphology** command in the script.

The dialog for that command appears and shows you that it will perform a 5x5 erosion and place the results into a new window. Click **Cancel** in the dialog. This script is made to operate on the Chromosomes image. You need to add an **Open** command in the script so that it will open the image before performing the rest of the commands.

3. Click once on the **Change Window** command at the top of the script.

4. From the **File** menu, select the **Open** command.

5. Find the folder that has the images and select the Chromosomes image and click **Open**.

Instead of opening the image, the **Open** command appears as the first item in the script. This is how you make a script. Just select menu commands as if you were performing the operation. When a script window is the active window, the script intercepts the command and places it into the script.

Now run the script.

6. Click on the **Run** button at the top of the script window. The script is self-running and it displays a series of alerts that tell you what it is doing. When it is finished, the script window comes back.

Notice the script left two windows on the screen. Let’s edit it to delete both of those windows and go into a loop that performs the same commands over again several times.

7. Scroll to the top of the script if it is not already there. Click on the first command which should be the **Open** command you inserted earlier.

8. Place a line label in your script by clicking on the **LABEL** button in the **Script Commands** window. In the dialog enter the word “Top” and click **OK**. You will see **Top** as the first line of your script. The underlined command indicates a line label, not a command. You use the labeled lines in scripts to control the order of execution, as you shall see in a moment.

9. Scroll to the bottom of the script and click on the **END** line.
10. From the **Window** menu, select **Dispose All Windows**.

**Dispose All Windows** closes all of the windows without asking you if you want to save them. Another handy command is the **Dispose Window** command. When you know that a certain window will be of no use from then on, in a script, you’ll want to use **Dispose Window** or **Dispose All Windows** to get rid of it instead of **Close Window**, which may ask you if you want to save.

11. Now we’ll insert a command which loops back to the top of the script. Click on the button labeled **LOOP** in the **Script Commands** window.

12. In the **to Label** pop-up selector, get the **Top** label. It should be the only one available. Enter 5 for **# of Iterations** to tell it to perform the same set of commands 5 times. Click **OK**.

![Loop Dialog Box](image)

13. Now get rid of the existing windows so the script does not get confused: Click on Chromosomes window and dispose of it. Click on the Corrected Image and dispose of it.

14. The script window and the **Script Commands** window should now be the only windows on the screen. Click on the **Run** button again. When you get tired of the script running, press the escape key at any time that there is not an alert showing on the screen. The script stops. Now let’s delete the changes we made to the script.

15. Again scroll down to the bottom of the script window.

16. Click-and-drag the cursor from the **Loop** command over the **Dispose All Windows** command you added previously. Both commands should be highlighted.

17. Click on the **CLEAR** button to delete these command lines from the script.

18. Scroll to the top of the script and click-and-drag the cursor from the **Top** label down to the **Open** command. When both commands are highlighted, click in the **CLEAR** again.

The only way to edit a script is to use the **CUT**, **COPY**, **PASTE**, and **CLEAR** buttons in the **Script Commands** window. If you try to use the commands with those names from the **Edit** menu, you will see the commands entered in the script instead of editing the script.

Likewise, the only way to close a script window is to use the go-away box in the upper left corner of the script window. Close the Background Subtract script now and answer **Don’t Save** to the **Save...?** alert. Also dispose of any windows which might be left over from running that script.
The next script we’ll look at is called Analyze Color.

1. Find the Analyze Color script and open it. Click on the Record button to turn recording on.

This script is made to work on the Color Image, which is supplied on the distribution CD. The Color Image is a 24-bit color image. This script uses the Split Color Channels command to separate that image into the Hue-Saturation-Value color space. Then it does Segmentation on the Hue to isolate the green leaf from the rest of the image, and sets up the measurements to compute the area and perimeter of the leaf.

After the Measure Segments command in the script is a command called Change Window. Recall that the Measure Segments command creates a new results window behind all of the existing windows. When you used Measure Segments interactively in an earlier tutorial, you brought the results window to the front by using the Window menu. To do this in a script, use the Change Window command to bring a window to the front (or move it to the back). Double-click on the Change Window command found after Measure Segments in the script. You see that the dialog names a window called Leaf Measurements which it will bring to the front. That window is created by the script and doesn’t even exist yet, but Change Window lets you enter its name at the time you are making the script.

Now let’s open the Color Image. As you already know, if you select the Open command from the File menu, that command will simply appear in the script. But you can still open an image, even while making a script:

2. Click on the Record switch and the indicator shows that the script is turned off. All the buttons in the Script Commands window are dimmed and clicking on the commands has no effect.

3. Now you can perform other commands from the menus without those commands going into the script. Find and open the Color Image file.

4. To use the script, click on the script window and click on the Run button to start it.

Again the script is self-running and displays alerts to tell you what it is doing. This script closes all of its intermediate windows before it finishes.
### 3.6.2 Allowing Dialog Interaction in a Script

Some commands, such as **Segmentation**, require user interaction with the dialog. We can build this into scripts very easily. For this example we will consider the texture segmentation algorithm which you learned in a previous tutorial.

1. Open the script called **Texture Segmentation**.

![Texture Segmentation Script Showing a + in the Dialog Column]

2. You will notice that there is a column in the middle of the script which has a “-” symbol for most of the lines, except the command line with **Segmentation** on it, which has a “+” symbol. This column indicates whether you want the dialog for that command to appear when it is reached. A “+” in this column tells the script that you do want the dialog to appear when this command is reached so the user can enter new values as the script is running. A “-” in this column tells the script not to display the dialog when this command is reached, and instead just use the dialog values you entered when you made the script. Some commands in a script, such as **Tile** in this script, have neither a “+” nor “-” symbol next to them. These commands do not have a dialog.

3. With script recording **OFF**, open the image called Curds&Whey. Then click on the script’s **Run** button.

4. When the script reaches the **Segmentation** command, the **Segmentation** dialog appears. You can then interactively segment the image by moving the upper threshold to about value 150 and **OK** the dialog. When you do, the script continues from the point at which it called the **Segmentation** command.

Close this script and images to continue with the next session.
3.6.3  Allowing Drawing Interaction in a Script

Sometimes, you want the user of a script to place an ROI or some other drawing item in an image before continuing with the next set of commands. In this session you will learn how to do that. We will use the Color Sharpen script. Find and open that script.

1. In the middle of the script, you see the command named **Alert**. This is placed in the script by using the **ALERT** button found in the lower part of the script window. Turn on Recording, and double click on the **Alert** line in the script to see its dialog.

2. This dialog allows you to compose a message to the user. It also has the option to continue the script, suspend the script or stop this script (and all scripts it may have been called from) at this point. **Cancel** this dialog so we can run the script and see how it works.
3. Turn off the script, open the Color Image, then run the script.

4. When the script reaches the **Alert**, it displays the message:

![Alert Image]

Message Displayed When the Script Reaches the **Alert** Command.

5. When the you click **OK**, you are free to create the rectangular ROI you wish to process. A temporary window opens up titled **Script Status**. The “Suspended” state indicates that you can make any needed changes and then continue the script from where it left off.

![Script Status Image]

The Temporary **Script Status** palette

After creating the rectangular ROI, click on the Continue button in the **Script Status** palette. The script will then complete.

Close the image and script without saving them to continue with the next session.
3.6.4 Attaching Scripts to Function Keys

IMPORTANT NOTE FOR MAC OS9 USERS:

In order to use the Function Key feature of *IPLab*, you must disable the Hot Keys feature of Mac OS9. The two will conflict and your function keys in *IPLab* will not work as described below if you do not disable the Hot Keys in OS9. To disable the Hot Keys in OS9, select the “Keyboard” control panel from the Control Panel sub-menu under the Apple menu. Click on the “Function Keys” button and configure the dialog as shown below. Note: On OS 9.0 the meaning of the checkbox is reversed. later versions look like this:

![Hot Function Keys Dialog Box in Mac OS9.1/9.2](image)

Sometimes, a script is so useful that you will want to use it many times. In that case it can be cumbersome to select the script from the Run Script dialog each time. Fortunately, you can create a one-button way to run such scripts, by assigning them to function keys on the keyboard. To illustrate this, we will use the Color Sharpen script from the previous example.

1. Go to the Edit menu and select the command Assign F Keys. If that command is dimmed, you probably have a script as the active window. Either turn the script off or close the script window.

2. When the dialog appears, click on the F9 line in the dialog. Then go to the Script menu and select Run Script (DO NOT use the Open Script command) from the menu.
3. In the **Run Script** dialog, find the Color Sharpen script and click **Run Script**. You see the name of the script appear in the **Assign F Keys** dialog next to F9. You may enter a different name for the script. This name will be used in the FKeys palette (discussed below).

![Assign F Keys Dialog](image)

**Assign F Keys** Dialog *After Setting the F9 Key*

4. **OK** the **Assign F Keys** dialog.

Now you can run the script by pressing a single button on the keyboard.

5. Close all of the open windows and **Open** the Color Image.

6. Now press the F9 key. That key invokes the Color Sharpen script, which asks you to select an ROI. Just select **Continue** from the script menu to finish running the script.
A palette window which lists all of the assigned F Keys is available. Choose **Show FKeys** command from the **Window** menu. (For this example we also assigned a command to FKey 1).

![FKeys palette with two assigned keys.](image)

You can click on the button in the window to perform the action, or use the keyboard as before. You can make the palette smaller by clicking in the zoom box (2nd square from the right edge of the title bar).

![FKeys palette after clicking zoom box.](image)

Click in the close box, or choose **Hide FKeys** in the **Window** menu to close the FKeys palette.

### 3.6.5 Querying the User with IPLab Variables

A script often needs specific information at the time that it is executed. For example, you may want to know how many times the user wants to execute the commands in a certain loop, or you may want to enter different parameter values into some command dialogs based on conditions that may change each time you run your experiment. This is where the **IPLab** variables come in handy. The variables array is an indexed list of 512 numbers which can be accessed by almost any dialog that accepts numerical parameters. There are several ways to set the values of variables while a script is running. In this session we’ll show you how to question the user and then use the **IPLab** variables to make decisions in the script based on the user’s answers.

Among the distribution scripts is a basic, do-nothing script called Query User which we’ll use as the foundation for the discussion. Find and open the script called Query User, and turn Recording on. We will examine this script in detail.

![Query User Script](image)

The first line is a label. You have already explored labels in a previous session of this tutorial. Labeled lines are often used as a place to jump or loop to. The next line is the **Enter Variables** command. This command is located in the **Script Commands** window.
Double-click the **Enter Variables** line in the script.

![Enter Variables Dialog Box](image)

**Enter Variables** Dialog Box.

This command is described more fully in the Scripting chapter, on page 143, but here we can see that it is set up to accept two numbers: The objective power (for a microscope) and the number of images to capture. When the script is run, the dialog appears in slightly modified form and the user is allowed only to enter numbers in the **Value** column. This information is placed into the specified indexes of the **IPLab** variables array (indexes 20 and 30). Cancel this dialog.

Next in the script is a label which is used as the top of a loop. This is followed by **New Image**. Here you would probably place a command which acquires an image. **New Image** is just a place holder for demonstration purposes.

Next is an **If** command. This command is placed into the script using the **IF** button in the **Script Commands** window. Double-click the **If** line in the script.

![If Command](image)

The **If** Command is Set to Jump to Label “16x” when Variable 20 Equals 16.

This dialog tests the value at variable number 20. If the value in that variable is equal to 16 then the script jumps to a location in the script with label “16x”. Cancel this dialog and notice that the script has a label “60x” a few lines down. If the jump is not taken then the script just falls through to the next line. We have placed here a label “60x” to make the situation a little clearer. Recall that variable number 20 was where **Enter Variables** stored the answer to one of its questions, namely the objective power. So using the **If** statement in conjunction with the **Enter Variables** command, we are able to conditionally execute portions of the script based on the user response at the time the script is run. In this case, the only thing that is done differently is setting the (x,y) units.
You can probably think of other things that you might want to do differently, based on the microscope objective you used. Those things would be included where the **Set XY Units** commands appear now.

This section of the script is finish by the **Loop** command. Double-click on the **Loop** command in the script.

![Loop Command Dialog](image)

**Loop Command Dialog** Getting the Number of Iterations from Variable 30

You see that the number of iterations is obtained from the variable number 30, where the **Enter Variables** command stored the user’s response to the number of images prompt. **Cancel** this dialog.

After the **Dispose All Windows** command in the script, you see the **Query** command. This one comes from clicking on the **Query** button in the **Script Commands** window. It is also used to get information from the user. Double-click on the **Query** command in the script.

![Query Dialog Box](image)

**Query Dialog Box**

This command allows you to program a question that the user will see when he runs the script. The question may have up to three responses which are encoded into three buttons. Only two of the buttons are used in this example. The number of the button chosen by the user is placed into the specified **IPLab** variable. This information is then used by the final command, If. Double-click on the final If command and you will see that it tests to see if variable number 10 has value 0 (which would indicate that the user pressed the button entitled “Another”), and if so, it makes the script jump back to the beginning of the script.

Before running the script, let’s open the **IPLab** variables window so we can see the values changing.

1. Turn OFF the script. Then select **Show Variables** from the **Window** menu.
2. Return to the script and click on the **Run** button to see how the script operates.

Once you have tried the script a few times, there is one enhancement we think you might want to consider. Let’s add a few lines to the script to keep track of the loop number within the loop. Sometimes the loop number is valuable in setting other parameters in analysis.
We’ll use variable number 5 to keep track of the loop index. The steps are that it must be cleared before the loop starts and it must be incremented each time through the loop.

3. Highlight the label **Loop Begin** in the script.

4. From the **Script** menu, select **Set Variable**. Set variable number 5 to value 0 and **OK** the dialog.

   ![Set Variable Dialog](image)

   **Set Variable** Dialog used the Clear the Variable Number 5

   4. Next highlight the command **New Image** in the script. This is the first command inside the loop, and we want to place a different command as the first command inside the loop.

   5. Again from the **Script** menu, select the command **Set Variable**. This time select the radio button that allows you to do arithmetic on the variable. You want to add the number 1 to the variable whose index is 5, so fill out the dialog as shown below, then **OK** the dialog.

   ![Set Variable Dialog](image)

   **Set Variable** Dialog Used to Increment the Variable Number 5 by 1

   That’s it. When you run the script you won’t see any difference, but the variable number 5 gets incremented. In this example we didn’t actually use the loop count from the **Variables** window anywhere else, but you could place inside the loop any command whose dialog can access variables to make use of this loop count.

   Close the script and all of the open windows to continue with the next session.
3.6.6 Advanced Script: Correlation Filter

In this example, we look at a script that performs a pattern matching operation using the Fourier Transform operation in *IPLab*.

The problem you will explore is the following: You have a certain pattern, or template, that may be found within some images. You want to find the exact location of that pattern. In this session you will simulate this situation by creating a template out of a portion of the Washington image. Then you will correlate the template with the original image so that there is a peak value in the correlation output at the location where your pattern is found.

The pattern matching algorithm you will use can be called a Phase Only Matched Filter. The mathematical formulation of the filter is as follows:

\[
D(f_x,f_y) = \text{FFT} \left( \text{Data}(x,y) \right)
\]

\[
T^*(f_x,f_y) = \text{Complex Conjugate} \left( \text{FFT} \left( \text{Template}(x,y) \right) \right)
\]

\[
G(f_x,f_y) = \frac{D(f_x,f_y) \cdot T^*(f_x,f_y)}{|D(f_x,f_y)||T(f_x,f_y)|}
\]

\[
\text{Output}(x,y) = \text{FFT}^{-1}(G(f_x,f_y))
\]

The result, \(\text{Output}(x,y)\), is in the image domain. There will be a peak at the location in the image where the template is found.

You have a script called Correlation Filter which implements this formula. Open that script using the Open Script command from the Script menu and scroll to the top.

![Correlation Filter Script](image)

Correlation Filter Script

This script is fairly complicated and worth looking at in detail.
After setting up the process, the **Transforms** command executes the FFT with Magnitude-Phase output on the active window. This gives the $D(f_x,f_y)$ factor. Then the script brings the Template window to the front and performs the FFT again. This gives $T(f_x,f_y)$. Now a mathematical trick is used to implement the correlation formula given above. The numerator, $D(f_x,f_y) \cdot T^*(f_x,f_y)$ can be performed by multiplying the magnitudes and taking the difference of the phases. The next several lines perform the phase difference calculation: Data Phase - Template Phase. The result is left in the Template.FFTPhase window. The formulas above show that the result must be divided by the magnitudes of the Data and Template transforms. This is equivalent to setting the magnitude of the result to 1. That is done next in the script with the **Set ROI Value** command. Then the inverse FFT is performed and the data is displayed. The next set of commands set up measurements to find the maximum value in the correlation plane, finished by **Measure ROI** to get the maximum value.

Once the magnitude of the peak is found, the **Set Variable** command is used to transfer that peak value to the **Variables** array (variable number 10, to be exact). Then the **Change Segment Color** command uses that variable to set the segment overlay color only at the peak. **Modify Segments** makes the red pixel bigger with a dilation. Finally, **Transfer Attributes** copies the segment overlay from the Correlation Plane window to the Data window.

### 3.6.6.1 Running one script from another

This script expects to work from two images. The pattern to search for (the “template”) is one, and the image in which to search (“data”) is the other. Further, the pattern must be prepared in a specific manner. To prepare the template window we will use a second script. Once it has prepared the template, it will run the Correlation Filter script to do its job.

Open the Set Up Correlation Filter script.

![Set Up Correlation Filter](image)

The Set Up Correlation Filter script.

This script defines an ROI on the Washington image, then allows the user to move the ROI to different locations within the image.
The script then copies the image data from within the ROI and pastes it into the center of a new window named “Template”. The **Barrel Shift** command (Math menu) is then used to prepare the pattern for use by the Correlation Filter script.

![Template Image before and after Applying the Barrel Shift Command](image)

Notice that the **Run Script** command is, itself, scriptable. This allows you to run one script from another. That, in turn, allows you to break up complicated scripts into a number of simpler scripts. The script being run does not have to be open, as long as it is available on disk. You can use up to 5 levels of scripts (that is, Script A calls B, which can call C, which can call D, which can call script E).

Once the Correlation Filter script completes, the Set Up script continues with the next instruction (Change Window). It closes some intermediate windows and tiles the remaining ones.

Now let’s run the scripts and see how well they work.

1. Since scripts are not commonly open when you run them, close both scripts, and any other images that are open.
2. Next, find and open the Washington image.
3. Use the **Run Script** command in the **Script** menu to run the Set Up Correlation Filter script.
4. When the script pauses, leave the ROI where it is. Just press **Continue** in the **Script Status** palette.

When the script completes, you will see a new window called Location and another, Correlation Plane. It may look bizarre, but it contains the information you want. In this image you’ll see a red rectangle. This is the location of the peak value in the correlation plane. The window called Location gives the x and y coordinates of this maximum. If you have done everything as described above, those values should read 85 and 115.

5. Activate the window Correlation Plane.
6. Select **Hide Segment** from the **Edit** menu to hide the red dot.
7. Move the cursor over to the x-y location (85,115) (use the **Status** palette to tell you where the cursor is) and notice that is where the bright dot is.
8. When you are finished, use the **Dispose All Windows** command in the **Window** menu to close all windows.
3.7 Tutorial 7: Point And Measure with Wand Tool

In this tutorial we will show you how to use the wand tool to point at and measure individual segments.

1. First, define what properties you want to measure. Select the Set Measurements command and check the boxes for Sum, Mean and Area only, and then OK the dialog.

2. Next, select the Measurement Options command. Fill in the dialog as shown in the figure below. Make sure the segment color is green and the outline color is red. Also, you must not check any of the boxes under the heading Ignore Segments Touching ROI At.

![Measurement Options Dialog Box](image)

3. Now open the image called Three Phase. It has a few clearly defined particles in it to measure.

4. Next, segment the image. Use the Segmentation command in the Analyze menu with the min. and max. parameters to 150 and 255, make sure that the color is green and OK the dialog. This leaves several green segments in the image.

5. Select the wand tool from the Tools window. Make sure the Segment/Draw selector in the Tools window reads Segment and that the color below that is also green.

![Selecting the Wand Tool](image)

6. Click on one of the segments in the image. You see the ROI snap to the outline of the segment. The wand tool places the ROI around the segment nearest your click point whose color you have set on the Tools window.

7. Now measure just this object using the Measure Segments command in the Analyze menu.
You can continue to measure individual segments by clicking on each one with the wand tool and then selecting **Measure Segments** from the **Analyze** menu.

The green overlay covers up your view of the image itself. Sometimes this is not convenient, you may want to see the image so you can distinguish more easily between the individual segments you want to measure and those you don’t. After the image is segmented, you can hide the segment overlay and still use the wand tool.

8. Go to the **View** menu and select the command **Hide Segment**. You see the green overlay disappear, but the segments are still defined.

9. Now continue to use the wand tool to select the segments you wish to measure. As you click in the image, the segment which is nearest and to the right of your mouse click point will be outlined by the ROI dotted outline.

If you use the **Measure Segments** command frequently on individual segments, it may be helpful to assign the **Measure Segments** command to one of the function keys on your keyboard. You can then invoke the command with a single keystroke instead of having to go to the menu every time. (You must have the extended keyboard with a row of F-keys along the top in order to do this step.)

10. With the image as the front window, go to the **Edit** menu and select the command **Assign F Keys**.

![Assign F Keys](image)

Assigning a menu command to an F-Key

11. Click on F9 with the mouse cursor then go to the **Analyze** menu and select **Measure Segments**.

12. **OK** the **Assign F Keys** dialog.

Now you can use the F9 key anytime you want to apply the **Measure Segments** command.
3.8 Tutorial 8: File Handling

You are no doubt used to the standard Macintosh file Open and Save As commands. Although these are convenient methods for interactive operation, the basic opening and closing commands require too much interaction to be useful when you want to process a large number of images. For processing many images, you would like something like a “batch mode” of operation. Say, “Here is a batch of files that I want to do the same thing to.” To address this issue in IPLab we have introduced two new methods of file input/output (file I/O): the file list and indexed files. We’ll just briefly discuss these issues here. Please pursue further details in the rest of the User’s Guide.

3.8.1 Indexed or Numbered Files

Suppose you are capturing many images in a session and you want to save them as sequentially numbered files. Indexed files were made for you.

1. From the File menu, select the command Set Indexed Name for Save.

Selecting the Command Set Indexed Name for Save

2. The dialog prompts you for a base name and the starting index you wish to use as well as a folder location where you with the files to be saved. As an example, the figure below shows you saving the images in the folder called Images. You can save the images anywhere you want, but you must specify the location using this dialog. When you are finished, click on the Set button.

Set Indexed Name for Save Dialog Box

3. Once this information is set, you can save images using the information. First you need an image to save: From the File menu, select the command New Image. Name it Untitled and make it 256x256, 1 frame, Byte type, then OK the dialog. This creates a new image for you.
4. Now select the **Save As** command from the **File** menu. When you select the option **Indexed File** at the bottom of the dialog, the name of the first indexed file appears next to that radio button. The folder information at the top of the dialog is ignored and the folder information you selected in the **Set Indexed Name for Save** dialog will be used instead.

![Image of the Save As command with Indexed File option selected.](image)

This set of commands is most useful in a script such as the following:

```
Set Indexed Name for Save
Set
Single Exposure
Save As
Loop
END
```

**Simple Image Acquisition Script Using Indexed Files for Saving**

You place the command **Set Indexed Name for Save** outside the loop. Then each time the **Save As** command is executed inside the loop, it bumps the index value by 1. The result is that you will have a set of files with names “z Series0000”, “z Series0001”, “z Series0002”, etc. If you want to try this script and you do not have a frame grabber or camera, try replacing the **Single Exposure** command with a **New Image** command.

The reverse operation can also be done. Once you save all those files, you can open them again using indexed files. First, use the **Set Index Name for Open** command, also in the **File** menu. Then use the **Open** command with the **Indexed File** option selected.
You can have only one name assigned to the base name for **Save**, one more for **Open**, and another one for **Delete File**. If you want to change base names within a script, you will have to use the **Set Indexed Name for Save/Open/Delete** command more than once. Since this dialog also wants the starting index number, you will not be able to enter just any number since this fixes the starting index at some value which you may not want. Instead, consider using one of the **IPLab** variables to hold the index. You might experiment with this using the script above, except, in the **Set Indexed Name for Save**, choose the starting index to come from variable #0 as shown in the figure below.

![Set Indexed Name for Save Dialog with the Starting Index Coming from Variable #0](image)

**Set Indexed Name for Save** Dialog with the Starting Index Coming from Variable #0

Now when the script is run, the variable #0 is accessed to get the index and is incremented each time through the loop.
3.9 Tutorial 9: Real Time Image Analysis

Sometimes you may want to analyze something on images as you are acquiring them, repeatedly updating the analysis results. This is done in *IPLab for Macintosh* with the aid of scripting. Here is a typical script you might use to watch the intensity values on the region of interest in an image:

![Image of script interface]

A Script for Real Time Image Analysis

The script begins by disposing of all existing windows. This is not required by the other steps, but makes it easier to run the script repeatedly. The script then acquires a full frame image, letting the user preview the capture for focusing and field of view selection. Next, an **Alert** is used to tell the user to draw an ROI on the image and select **Continue** from the script palette. After this is done, the large window is moved aside and another image is acquired. This time, the image is only the size of the ROI that the user has selected. Capturing on a smaller portion of the image makes the process somewhat faster. This last captured window is named “Data”. All future captures are placed into this window. It is also moved aside. Next, **Set Measurements** is used to set up what we want to measure. This should be a single parameter, because we want to collect the data into a column oriented results table and plot the intensity versus frame number. In this example script, the mean intensity is measured. The **Measure ROI** command creates a results window, which is then made into the active window by the **Change Window** command. This is so we can setup the QGraph options and then view the results as QGraph. When this is done, the Data window is brought back to the front and the **Status** and **Tools** palettes are hidden. Finally, the script enters a short loop in which an image is captured and the ROI is measured.

This script can be used as a starting place to perform other kinds of real time image analysis. While this script measures the mean intensity through the **Measure ROI** commands, you can replace this type of analysis with one of several other types. You can of course use **Measure Segments** in place of **Measure ROI**. This would allow you to measure and plot the intensity values at more than one location on the image, at the cost of being a little slower.

You can also:

- Plot real time histograms,
- Do real time Fourier analysis,
- Extract and plot a slice or line of data through the image in real time, and
- Get a real time plot of a row sum or column sum of the ROI.
4 Should-Read Section

The information in this chapter is very useful, but not crucial to your use of IPLab. Feel free to start using IPLab without reading this chapter, and then return and read this chapter to become a more powerful IPLab user.

Chapter 2, the Must-Read chapter, describes the most important parts of the IPLab environment. Chapter 5, the Operation chapter, details the inner workings of IPLab.

4.1 # / V Buttons, L / SV Buttons

These buttons let you use variables for the dialog box's parameters. You will see these buttons in IPLab dialogs next to text-entry boxes.

The # / V button: # stands for number, and V means numeric variable. Click on the # button to change it to a V, and vice versa.

The # symbol means that this value is a simple number, the number 216.

The V symbol means that this is the index to a numeric variable. Variable #216 holds the value that the dialog box will use.

Variables and their buttons are described in much more detail in the IPLab Variables section, below.

4.2 IPLab Variables

4.2.1 What Are IPLab Variables?

Variables are storage places for values used in commands. For example, you can store the values for a new image’s width and height (for example, 640 pixels and 480 pixels) in two variables. You can then use those two variables in the New Image command (File menu) to make a new image with that size (640 x 480).

You will almost always use variables within scripts. Scripts often require specific information at the time that they are executed. For example, the script may ask the user what image width or which filter wheel to use. The script would store the user’s response in a variable. When it came time for the script to create an image or move the filter wheel, it would use the stored response. This IPLab Variables section describes IPLab variables and how to use them.

The main use of variables is in creating a custom user interface in scripts. First, your script should set the values of some variables. Then it should use those variables to set the values of parameters in dialog boxes. This provides a simple way to change key command parameters without the need for the user to interact with each dialog. For example, you may assign a certain value to variable #5, and then use that variable as a new image's height. Then, you could alter the image height for future images simply by changing the variable, without having to show the New Image dialog to the user.

It is very useful to remember that the exposure time used in the last acquisition is always stored in variable #255.
4.2.2 Numeric Variables

Numeric variables, as described above, are storage places for numbers used in IPLab commands. Whenever the IPLab manual refers to the generic term “variables,” it is talking about numeric variables. The numeric variables and their values can be found in the Variables window, described on page 28. To see this window, choose the Window menu command Show Variables.

4.2.2.1 Setting Numeric Variables

You can use the Set Variable command in the Script menu to manipulate the values of any numeric variable, even if the Variables window is hidden.

Set Variable Dialog Box

For the sake of example, open the Set Variable dialog and type the variable number 5. Enter the number 480, as shown above. In this example, you are setting variable #5 to equal 480, which will be used as the height of a new image in the next example. You can read more about how to use the Set Variable command in the Script section of this manual (page 291). Also, please note that there are other ways to assign values to numeric variables, including the immensely helpful Enter Variables command, found in the Script Commands palette, described on page 143.

You can also set variables when the Variables window is showing by selecting a range of variables and using the Set Value command (Edit menu).
4.2.2.2 Using Numeric Variables in Dialogs ( # / V )

Most command dialogs that accept numerical values have special symbols next to the boxes where you enter the values.

![Example of #/V Toggle Buttons](image)

These # and V icons represent either number or variable. You toggle the symbol between # and V by clicking on it. If the # symbol is showing, then the value in the corresponding text edit box is used directly for that parameter in executing the command. If the V symbol is showing, then the value in the corresponding text edit box indicates a variable number. When the command is executed in this case, the value of the variable is used for that parameter. The example figure shown above is the Custom Size section of the New Image dialog box. The width of the new image will be the number of pixels given: 640. The height of the new image will be taken from variable number 5.

Again for the sake of example, choose the New Image command from the File menu. Click the Custom Size radio button. Type 640 for the Width, and make certain that the button beside the Width field has a # symbol on it. Next, click on the button beside the Height field, so that it reads V. Now type the number 5 in the Height field. You already set variable #5 to equal 480 in the previous example, so when you click OK in the New Image dialog, the new image’s size will be 640 x 480 pixels.

4.3 Keyboard Keys

4.3.1 Option Key

You can use the Option key to change the shape and size of ROIs and drawing objects. When you hold down the Option key, four handles appear at the selection's corners. Clicking and dragging these handles moves the sides of the ROI.

Moving, resizing, and reshaping an ROI is described in detail on page 13. Basically, just hold down the Option key while an ROI tool is selected. Click inside the ROI and drag it to move it. Click and drag on one of the handles to change its size or shape.

4.3.2 Arrow keys

Use the arrow keys on the keyboard to move the ROI within the active image. When a drawing tool is selected, you can press the arrow keys to move any selected drawings.

You can also use the arrow keys to position the paste region after you use the Paste command and while the paste region is still “live.” This ability gives you control over where the pasted image will sit.

The Command (⌘) key and the arrow keys will move you through sequences. When the data in the active window contains a Z-sequence, hold down the Control key and use the up/ down arrow keys to change the Z-plane displayed by the image window.
4.3.3 Command-Period and Escape Key

Generally, the Command-period key combination (when you hold down the command key and press the period key) and the Escape (esc) key perform the same function, to signal “Cancel” or “Stop”.

You may use these keys to:
• Cancel from any dialog, exiting without making any changes
• Cancel the creation of polygonal and line ROIs, drawings, and segments
• Stop using the Measure Angles and Measure Lengths tools
• Cancel pasting after you have and used the Paste command but while the paste region is still "live" and floating above the image
• Stop scripts that are running

4.3.4 Delete Key

The action of the Delete key depends on which tool is selected on the Tools palette:
• When any drawing tool is selected on the Tools palette, pressing the Delete key deletes the selected drawings.
• When any Segment tool is selected, pressing the Delete key clears selected segments of all colors.
• When the Registration Mark tool is selected, pressing Delete removes the selected registration marks.
• When any ROI tool or any of the other image tools is selected, pressing the Delete key clears the selected image data just as if you had picked Clear from the Edit menu.

Pressing Delete also undoes pasting of 2D data before the process is completed.

4.3.5 Shift Key

The Shift key constrains drawing for the ROI, segment, and drawing tools. You can create ROIs, segments and drawing objects which are square or circular by holding down the Shift key while using the rectangular or oval tools, respectively. The Shift key also constrains the polygon and the line tools to 90° and 45° angles.

While viewing data as text, you can use the Shift key to select rectangular regions of text. Click once on a cell, hold down the Shift key, and click on another cell. This selects a rectangle of data having these two points as its corners. In a script window, you can also hold down the Shift key to select a range of commands.

Finally, if you hold the Shift key down while starting IPLab, the IPLab extensions will not be loaded.

4.3.6 Quit without Saving: Command-Option-Shift-Quit

If you hold down the Command, Option, and Shift keys all at once, and then choose Quit from the menu, or type a “Q”, IPLab quits immediately, without saving any files. Be sure that you have already saved any important windows before using this feature.
4.4 Batch File Processing

**IPLab** allows you to apply your lab protocols interactively, to one image at a time, so that you can see the effects immediately. However, you can also apply your protocols to multiple images as a batch. This is accomplished by using indexed files, file lists, and scripts.

Indexed files are files that have a base name and a number, which help to organize them. You set up indexed files using the Set Indexed Name commands (File menu). Then you can control those files using the Indexed File options in the Open, Save As, and Delete File commands. For more information on indexed files, see the description for the Set Indexed Name command on page 172.

File lists are just what they sound like: lists of image files. You can create file lists with the Edit File List command (File menu). You can then use the Select File List command to pick an appropriate list. You can control that list's files by using the File Lists options in the Open, Save As, and Delete File commands. These commands would obtain the names and locations of the files from the named file list instead of from their dialogs. For more information on file lists, please see the Edit File List description on page 169.

Indexed files and file lists let you manage a large number of files one at a time. To batch process files, you would use indexed files or file lists to open the images within a script, which would then process the data. Basically, all you need to do is:

1. Record an Open: from File List or Open: Indexed File command near the beginning of a script, and
2. Record a Loop command at the end of the script to loop back to the image-opening command.

Each time the script runs through the loop, it will automatically open and process the next file. The section starting on page 135 describes scripting in detail.

4.5 **IPLab** Text Files

This section describes **IPLab** text files so you may import and export text files to and from **IPLab**.

4.5.1 Importing Text Files

Two types of text files are handled: those which contain row labels, and those which do not. Both files have a simple “header” followed by the data.

**No row labels:** For a text file without labeled rows, **IPLab** expects this format:

<table>
<thead>
<tr>
<th>integer width</th>
<th>integer height</th>
</tr>
</thead>
<tbody>
<tr>
<td>“I” or “F”</td>
<td>(Integer or Floating Point data --described below)</td>
</tr>
<tr>
<td>A blank line</td>
<td></td>
</tr>
<tr>
<td>data(0,0)</td>
<td>data(1,0)</td>
</tr>
<tr>
<td>data(0,1)</td>
<td>data(1,1)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>data(0,height-1)</td>
<td>data(1,height-1)</td>
</tr>
</tbody>
</table>
With row labels: For a text file with labeled rows, IPLab expects this format:

integer width
integer height
“IR” or “FR” (Integer or Floating Point data --described below)
In this row, column labels are saved with a tab as the first character

row label string     data(0,0)     data(1,0)     ...     data(width-1,0)
row label string     data(0,1)     data(1,1)     ...     data(width-1,1)
...                   ...                        ...
row label string     data(0,height-1) data(1,height-1) data(width-1,height-1)

For either type of text file, all of the entries are ASCII. The width and height parameters tell IPLab how much data to read and the dimensions of the resulting data window when performing the Open command. These are integers (they do not have a decimal point) and are positive (greater than 0). The type flag (in the third row) tells the Open command whether to place the data into a long integer or a floating-point image. The file uses either an ASCII “I” to designate integer type or an ASCII “F” to designate floating point. If the file includes row labels, “IR” or “FR” must be used to designate integer or floating point. If you designate that the data is integer type by using “I” or “IR” in the header, the file is read into an image of type long integer. If you wish to make the data byte or short integer type after opening the file, use the Change Data Type command from the Math menu. If there are no column labels, the next row should be blank, but it must be present.

If the file uses “IR” or “FR” in the data type field, IPLab expects to see row labels at the beginning of each row of data, followed by a tab character. Each row after the first three (width, height, I or F) must be terminated with a tab, including the row containing the row labels.

The data samples themselves should be tab delimited, although IPLab will accept space-delimited values also. Since the dimensions of the data come from the first two lines of the text file, IPLab ignores carriage returns and line feeds. Integer data must not contain decimal points.

Note: It is often helpful to cut text files into small pieces in order to open them quickly in IPLab.

4.5.2 Exporting Text Files

One of the most common questions is, “How do you export text files to Microsoft Excel?” This is actually quite easy. Use similar steps when exporting to any other spreadsheet program:

1. First, save the data as text.
   a. Select Save As from the File menu, and
   b. In the Format box, choose Text. The file is now a text file with the format described above.

2. When you open the file in Excel, the Text Import Wizard will automatically appear. The default settings will open the file correctly. The original data type is Delimited. You can start the import at any row you like, but row 5 (the first data row) is good, and the file origin is Macintosh. IPLab text files are tab delimited.

   If you like, you can click the Finish button immediately, without going all the way through the Text Import Wizard.
4.6 Data Views

There are several ways to view data:

**Image View**

This is the common, pictorial display of your data.

**Text View**

The text view displays all the image data as numbers in a spreadsheet-styled window. Each cell in this spreadsheet contains the intensity value(s) of the corresponding pixel in the image. Instead of being represented by a color or shade, the pixel is now represented as a number.

**QGraph View**

The QGraph view displays a graph or plot of up to 10 columns from the data. Use the View Options command (View menu) to set up your graph.

**Perspective View**

The perspective view displays 2D data as 3D, using the pixel intensity as the third dimension. IPLab pseudocolors the perspective display to help distinguish the different altitudes (pixel intensities). Only the 2D data that was within the ROI will be displayed, letting you focus on interesting features.

*IPLab* only presents QGraph and perspective views for grayscale (actually, indexed color) images.

Change the way the current window is displayed by using the View As command (View menu). To alter the views, use the corresponding commands in the View Options hierarchical command (View menu).

All of these views are described in more detail on page 205, in the View As command’s section.

4.7 Saving *IPLab* Settings

Each time you quit the *IPLab* application, it saves the current parameters for all commands in the Command Data file.

The Command Data file is in the Preferences folder:

IPLab 3.6 Folder: IPLab 3.6 User Folder: IPLab Preferences: Command Data

The next time you start *IPLab*, it reads the contents of this file and restores from it each command’s parameters. This removes the need for you to constantly re-enter values that you often use. If the file does not exist, *IPLab* assigns default settings to each command.
## 5 Operation

### 5.1 File Formats

*IPLab* allows you to save and read images in a variety of file formats using the commands **Open**, **Open As**, **Save**, and **Save As**, which are all found in the **File** menu. The file format will affect what information is saved along with the image data. It also affects how easily you will be able to view the saved image in other programs.

### 5.1.1 Descriptions of File Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPLab:</td>
<td><em>IPLab</em> can open and save data as its own file format, referred to as IPLab format. The Appendix describes the IPLab format's structure on page 343.</td>
</tr>
<tr>
<td>TIFF:</td>
<td><em>IPLab</em> can open and save 2D images and 3D image sequences as TIFFs. For saving and reading 2D data, <em>IPLab</em> supports most common TIFF (Tagged Image File Format) formatted files. <em>IPLab</em> can read most non-compressed TIFF files that conform to the TIFF 6.0 specification. This includes monochrome 8-bit, 16-bit signed and unsigned, 32-bit signed, and floating point images; and color images (8-, 24-, and 48-bit color). <em>IPLab</em> does not support compressed TIFF files. For saving and reading 3D data, <em>IPLab</em> uses the multiple image feature of the TIFF format to store each frame.</td>
</tr>
<tr>
<td>PICT, PICT (QT):</td>
<td><em>IPLab</em> can open and save data as the Macintosh PICT format. It also can open and save PICT (QT) files saved with QuickTime compression. These are the only compressed images that <em>IPLab</em> will work with. Please note that compression can alter data.</td>
</tr>
<tr>
<td>Text</td>
<td><em>IPLab</em> can save data as tab-delimited text by converting the binary data to ASCII, which can then be viewed in spreadsheets and other programs such as Microsoft Excel. Please see page 123 for the exact format of the <em>IPLab</em> Text file.</td>
</tr>
<tr>
<td>FITS</td>
<td><em>IPLab</em> opens and saves the FITS (Flexible Image Transport System) format. <em>IPLab</em> attaches a FITS header to images when you save them in this format. You can then use the <strong>Edit FITS Header</strong> command (<strong>File</strong> menu) to view the stored information. Save your FITS data after acquisition to store the acquisition information in the FITS header.</td>
</tr>
<tr>
<td>EPR</td>
<td><em>IPLab</em> opens and saves the EPR and PSF formats written by the Scanalytics deconvolution program, <em>Exhaustive Photon Reassignment</em>. You will have to add the &quot;.epr&quot; and &quot;.psf&quot; extensions yourself, of course. (The EPR and PSF formats are the same format.)</td>
</tr>
<tr>
<td>Raw</td>
<td><em>IPLab</em> can save data as raw data, meaning that it saves the binary pixel data without any information about the image (such as width, height, data type, <em>etc.</em>). To open raw data, define its format with the <strong>Foreign Formats</strong> command (<strong>File</strong> menu), and then use the <strong>Open As</strong> command (<strong>File</strong> menu).</td>
</tr>
</tbody>
</table>
5.1.2 Data Saved in Each File Format

Which file format should you choose? The IPLab format retains almost all of the information you may have entered in an image. Single plane TIFFs are widely supported, especially with Byte type data. Text format is ideal for exporting measurement results to another program, such as Microsoft Excel. EPR format is used when exporting data to the Scanalytics deconvolution program, Exhaustive Photon Reassignment. Of these file formats, only the QuickTime-compressed PICT (PICT-QT) format loses image data (due to the compression). Saving an image in any format besides IPLab can lose some information in the form of overlays, CLUTs, labels, and other parameters.

The following table indicates how images are handled by the different file formats. The last column, Parameters, represents the Normalization and units parameters.

<table>
<thead>
<tr>
<th>Data Types</th>
<th>Sequences</th>
<th>Overlays</th>
<th>CLUT</th>
<th>Column and Row Labels</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPLab</td>
<td>All</td>
<td>Yes</td>
<td>Saved</td>
<td>Saved</td>
<td>Not saved</td>
</tr>
<tr>
<td>TIFF</td>
<td>All</td>
<td>Yes</td>
<td>Stamp onto image first</td>
<td>Saved</td>
<td>Not saved</td>
</tr>
<tr>
<td>PICT and PICT (QT)</td>
<td>Byte and Color 24</td>
<td>No</td>
<td>Stamp onto image first</td>
<td>Saved</td>
<td>Not saved</td>
</tr>
<tr>
<td>Text</td>
<td>All except Color 24 or 48</td>
<td>Yes</td>
<td>Stamp onto image first</td>
<td>Not saved</td>
<td>Saved</td>
</tr>
<tr>
<td>FITS</td>
<td>Byte Short Integer Long Integer Float</td>
<td>Yes</td>
<td>Stamp onto image first</td>
<td>Not saved</td>
<td>Not saved</td>
</tr>
<tr>
<td>EPR</td>
<td>Unsigned 16</td>
<td>Yes</td>
<td>No</td>
<td>Saved</td>
<td>Not saved</td>
</tr>
<tr>
<td>Raw</td>
<td>All</td>
<td>No</td>
<td>No</td>
<td>Not saved</td>
<td>Not saved</td>
</tr>
</tbody>
</table>

The IPLab format is the only one that will save segment and drawing overlays with the image. IPLab files keep the image and overlay data separated in the file so your data will not be affected by the overlays. To export drawing-annotated images to another software package through any other file format, first use the command Stamp Overlays on Image (Edit menu) to make the overlays a permanent part of the image data.

FITS and Text files do not keep a CLUT, so if you save pseudocolored images in these file formats, you will lose the coloring information.

You cannot save 1D data in PICT or EPR formats.

Note: Numerical measurement results may contain column and row headings. To save these headings, you must save the results as Text or IPLab format.

5.1.3 Other Formats

To open IPLab-format files in IPLab for Windows, append “.Mac” to the end of the file name. IPLab will open IPLab for Mac files in IPL format (our Windows equivalent of the IPLab format).

IPLab also allows you to read foreign image files, including image files which may have been created under other operating systems. You should use the Foreign Formats and Open As commands (both from the File menu) to open files of type Foreign. The Foreign Formats command is described on page 173, and the Open As command is detailed on page 159.

Script files have a special format which can be read only by IPLab. Scripts are saved using the Save As button within the Script Commands palette (as shown on page 135).
5.2 Data Types

Image data may be one of seven data types. These data types determine how much and what range of information can be stored in each pixel.

**Byte:** 1 byte (8 bits) per pixel; values in the range [0, 255]

**Short Integer:** 2 bytes per pixel; values in the range [-32768, 32767]

**Unsigned 16:** 2 bytes per pixel; values in the range [0, 65535]

**Long Integer:** 4 bytes per pixel; values in the range [-2^{32}, 2^{32}-1]

**Floating:** 4 bytes per pixel; values approximately in the range [-3.4E38, 3.4E38]

**Color 24:** 4 bytes per pixel; with 8 bits assigned to the intensity of each color: Red, Green, and Blue

**Color 48:** 6 bytes per pixel; with 16 bits assigned to the intensity of each color: Red, Green, and Blue

The same data type represents all of the pixels in an image window.

The first five data types (byte, short integer, unsigned16, long integer, and floating) are referred to as *indexed color types*. Indexed color data types rely on a Color Look Up Table, or CLUT, to convert data values into displayed colors.

The last two types, color 24 and color 48, are referred to as *direct color types* because they can be displayed directly without a color table. The concepts of color tables and direct display are explained in the following sections.

5.2.1 Indexed Color

5.2.1.1 Color Look Up Tables

Indexed color type files (described above) require a correspondence table to translate the data values into displayed colors. This table is the Color Look Up Table, or CLUT. The CLUT is a table which has 256 entries. In each entry, there is a set of three numbers which correspond to red, green, and blue (R,G,B). Each of these numbers can have a value between 0 and 255. Each indexed color image window has a CLUT assigned to it as part of its data.

5.2.1.2 Displaying Indexed Color Images

When an image is displayed, each pixel value in the image is represented by an entry in the CLUT, and the R, G, and B numbers in that location of the CLUT are sent to the display electronics. The display electronics use these values to control the relative intensities of groups of red, green, and blue dots at each pixel location on the display.

By mixing various combinations of R, G and B, you can potentially display $2^{24}$ possible colors. However, you can not utilize more than 256 combinations of red, green and blue intensities at one time, because there are only 256 entries in the CLUT. If the CLUT is constructed so that each R-G-B entry in the table has R = G = B, then only black, white and grays can be displayed. (Equal measures of red, green, and blue light make gray.) If R = G = B = index value (“row number”) of the CLUT, then the grayscale intensity on the display is proportional to the pixel value in the image, as black for 0-valued pixels and white for 255-valued pixels.

In order to vary the displayed color or grayscale intensity for pixels in an indexed color image, you can change the value of the pixel data, which changes the CLUT entry used by that pixel.

*IPLab* performs display normalization to transform the raw data values into values in the range 0-255. This process sends the normalized data values to the CLUT for display. *IPLab* performs display normalization without destroying the original values of your data.
Options for display normalization are provided in the **Normalization** command found in the **Enhance** menu. If you have a pseudocolor CLUT attached to an image, most changes in the **Normalization** command will also affect which displayed colors are assigned to which data values.

### 5.2.1.3 Built-In CLUTs

**IPLab** gives you direct access to the CLUT associated with each image window. You can change the CLUT associated with the active data window by using the **Pseudocolor** command in the **Enhance** menu. **IPLab** provides you with several color tables from which to choose.

In the **Monochrome** CLUT, as the value of an image pixel increases from 0 to 255, the image intensity increases from black through the gray levels up to white. In the **Reverse Mono** CLUT, the image intensity decreases through the gray levels as the pixel value increases. The other CLUTs provide you with simple options for pseudocolor (also called false color) image enhancement. The other CLUTs are described in the Enhance section of the Menu Reference chapter, under the **Pseudocolor** command (on page 214).

### 5.2.2 Direct Color

The last two data types, color 24 and color 48, contain enough information within the pixel values themselves to display the image data in color. These data types are called direct color types because no CLUT is required when they are displayed. In color 24 image data there are four bytes assigned to each pixel (three bytes for data and one unused). Each pixel in a color 24 image actually contains three values. These three values correspond directly to the intensities of the red, green and blue components (RGB) of the pixel’s color. It is as if three different images were contained within the one color 24 image, each one controlling one of the three basic color components. We call these three sub-images the **components** of the color image. The figure below indicates how the red, green and blue components are stored in one color 24 pixel.

<table>
<thead>
<tr>
<th>31 0</th>
<th>31 0</th>
<th>31 0</th>
<th>31 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused</td>
<td>Red</td>
<td>Green</td>
<td>Blue</td>
</tr>
</tbody>
</table>

The Bits of a Color 24 Pixel

If the shaded bits represent 1’s and the open bits represent 0’s, then the red, green and blue components represented by this color 24 pixel are (168,42,21).

Color 48 pixels are similar to color 24 pixels, but each color component is represented as a value of type unsigned 16. This means that each color component in color 48 images can have thousands of intensity levels which gives much better control over display quality.

You may think of the red, green, and blue values of color pixels as parts of a 3D vector. Then, by transforming the vectors for each pixel, you can arrive at many alternative ways of representing the color information in each pixel. These transformed coordinate systems provide other ways of looking at the data, which may improve your ability to extract information from the image. **IPLab** directly supports five of the most commonly used color coordinate systems:

- **RGB** Red, Green, Blue
- **YIQ** Y (Intensity), In-phase, Quadrature
- **HSV** Hue, Saturation, Value
- **CMY** Cyan, Magenta, Yellow
- \( R, G, B \) Percentage light from each of red, green, and blue (see description on page 133)

Each of these color coordinate systems is discussed more fully below. You can split a color image into the components for any of these coordinate systems. You can merge RGB, YIQ, HSV, or CMY components to form a color image. In the descriptions below, we provide the algorithms for transforming from RGB into each of the other coordinate systems.
Using the **Affine Color Transform** command, you can form other custom color systems as well.

### 5.2.2.1 RGB

RGB, of course, stands for Red-Green-Blue. This is the default color coordinate system. It corresponds directly to the red, green and blue components used by the computer to display each pixel. RGB cameras also exist which provide these three components directly as outputs. Certain color frame grabbers can accept these inputs to allow you to bring color images into your computer as live video.

### 5.2.2.2 HSV

The Hue-Saturation-Value coordinate system is a nonlinear transformation of RGB. HSV corresponds more closely to the way humans perceive color than the other color systems do. Hue ranges in value from 0 for red, through 85 for green, through 170 for blue, to 255, which corresponds to a very reddish violet. Saturation tells how much white is mixed with the color. A value of 255 for Saturation provides a “pure” hue. As Saturation decreases to 0, more and more white is mixed with the color. (Keep in mind that mixing with white means mixing equal amounts of all the primary colors.) The Value component of HSV provides intensity information. Various intensities of gray can be obtained by setting the Saturation component to 0 and varying the Value component.

The following algorithm is used to transform RGB coordinates into HSV coordinates:

\[ V = \max(R, G, B) \]
\[ S = \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B)} \]
\[ RC = \frac{\max(R, G, B) - R}{\max(R, G, B) - \min(R, G, B)} \quad \text{measures distance to red} \]
\[ GC = \frac{\max(R, G, B) - G}{\max(R, G, B) - \min(R, G, B)} \quad \text{measures distance to green} \]
\[ BC = \frac{\max(R, G, B) - B}{\max(R, G, B) - \min(R, G, B)} \quad \text{measures distance to blue} \]

if \(R = \max(R, G, B)\) then \(H = BC - GC\)

if \(G = \max(R, G, B)\) then \(H = 2 + RC - BC\)

if \(B = \max(R, G, B)\) then \(H = 4 + GE - RC\)

Once \(H, S,\) and \(V\) are computed by these formulas, they are scaled so that they are in the range \([0, 255]\).

### 5.2.2.3 YIQ

This coordinate system is widely used in television transmission. The Y component contains the intensity information in the image, while the in-phase and quadrature components (I and Q) carry the color information. This system is derived from RGB by a linear transformation.

The standard linear transformation from RGB to YIQ is:

\[ Y = 0.299R + 0.587G + 0.114B \]
\[ I' = 0.596R - 0.274G - 0.322B \]
\[ Q' = 0.211R - 0.523G + 0.312B \]
Notice that \( I' \) and \( Q' \) may actually have negative values (the reason for the primes will become apparent below). But not all possible \( YI'Q' \) triplets can represent a legitimate image. To see this, consider the inverse transformation; for example, \( YI'Q' \) to RGB:

\[
\begin{align*}
R &= 1.0Y + 0.956I' + 0.621Q' \\
G &= 1.0Y - 0.272I' - 0.647Q' \\
B &= 1.0Y - 1.106I' + 1.703Q'
\end{align*}
\]

You will notice from this inverse transformation that some \( YI'Q' \) triplets could yield negative values for \( R, G \) or \( B \). Since \( R, G \) and \( B \) must always be positive, this means that some values of the \( YIQ \) triplet are not really allowed. Therefore, \( YI'Q' \) is an unlikely coordinate system to use when synthesizing color images.

So the \( I \) and \( Q \) components may actually have negative values, which cannot be represented in Byte type. In order to remain consistent with the Split Color Channels and Merge Color Channels commands, it is necessary to scale and offset these components so that their values are in the range \([0,255]\). The actual transformation from RGB to YIQ used in IPLab is as follows:

\[
\begin{align*}
Y &= 0.299R + 0.587G + 0.114B \\
I' &= \frac{0.596R - 0.274G - 0.322B}{2*0.596} + 128 \\
Q' &= \frac{0.211R - 0.523G - 0.312B}{2*0.523} + 128
\end{align*}
\]

If you expect to use the \( YIQ \) coordinates on a regular basis, you may want to work with integer-valued component images. You can use the Split Color Channels command to create the Byte type \( YIQ \), and then convert these components into the classical \( YI'Q' \) defined above. You can then start with \( YI'Q' \), perform a transformation to the Byte type \( YIQ \), and then merge the coordinates into a color image.

### 5.2.2.4 CMY

In some earlier versions of IPLab for Macintosh, and in most art-based image processing packages, the Cyan-Magenta-Yellow coordinate system has been used to represent a subtractive color coordinate system. In that case, yellow, for example, was represented as the absence of blue. While this is useful for printing, it is not useful for most scientific applications. For this, we recognize that pure yellow is formed as an average of red and green. Thus, in order to tell how much yellow is in a pixel, we look for the combination of red and green. Cyan and magenta are similarly defined as mixtures of green and blue, and red and blue, respectively. This leads us to the following definitions for CMY: the transformation of RGB to CMY is given by:

\[
\begin{align*}
C &= \frac{G + B}{2} \\
M &= \frac{R + B}{2} \\
Y &= \frac{R + G}{2}
\end{align*}
\]
Values of C-M-Y defined in this way are guaranteed to be between 0 and 255. This is important because most often we capture images in RGB and we need to transform to CMY only to analyze the colors. The inverse transformation is defined as follows:

\[
\begin{align*}
R &= -C + M + Y \\
G &= C - M + Y \\
B &= C + M - Y
\end{align*}
\]

The results are then clipped to the range 0-255 so that the RGB images will be in the range of Byte data.

If you want the alternative definition of CMY (C’M’Y’ as used in printing color separations), you can get it by the following method:

1. Split the color image to its RGB components,
2. Apply the Invert command found in the Enhance menu, and
3. Invert only the data values on each component.

Then:

\[
\begin{align*}
C' &= 255 - R \\
M' &= 255 - G \\
Y' &= 255 - B
\end{align*}
\]

5.2.2.5 \( R_s, G_s, B_s \)

The \( R_s,G_s,B_s \) color space is also referred to as XYZ. This is a normalized color space which ignores the effects of brightness. It is tremendously useful for examining histology sections and similar images. The transformation from RGB to \( R_s,G_s,B_s \) is:

\[
\begin{align*}
R_s &= 255 \times \frac{R}{R + G + B} \\
G_s &= 255 \times \frac{G}{R + G + B} \\
B_s &= 255 \times \frac{B}{R + G + B}
\end{align*}
\]

This transformation is not reversible; \textit{i.e.} you cannot recover RGB if all you know is \( R_s,G_s,B_s \).

5.2.2.6 Displaying Direct Type Images

In the unlikely case that your display card displays only 256 colors (or grays), or if you are operating in the 256 colors mode, then \textit{IPLab} will display your images as well as possible using the colors it has available to it. This means that in order to display a color 24 or color 48 image, some CLUT must be used when displaying on an 8-bit monitor. Therefore, in \textit{IPLab}, even color 24 and color 48 images have a CLUT attached to them. In many cases, this CLUT will be the System CLUT, since it has a very good color distribution that permits most images to be represented fairly well. But you may attach any CLUT you wish to your images using the \text{Pseudocolor} command (Enhance menu). The advantage in this is that you can adjust the CLUT to provide a better-looking display on an 8-bit monitor.

The \text{Change Color Type} command allows you to convert your 8-Bit color images into Color 24 or Color 48 images. When you convert an 8-bit image to a color 24 or color 48 image, the CLUT of the original indexed type image stays with the new direct type image so the image displays properly on an 8-bit monitor.
5.2.2.7 Dithered Display of Direct Type Images

There are actually two ways to display Direct type images on an 8-bit monitor: **Dither On** and **Dither Off**. These display options are selected from the **View Options: Image** command (View menu). With **Dither Off**, each pixel is displayed in the color from the current CLUT which most closely resembles the true color of the pixel. Of course, the displayed color may be different than the true color if the CLUT does not have the true color in it. The color mismatching that occurs in such cases can make your images look “contoured.”

With **Dither On**, such display errors are taken into account as the image is displayed. Dithering tries to adjust the displayed colors of adjacent pixels to compensate for the display errors. Dithering often provides a better-looking image by fooling the eye into thinking that more colors are displayed than are really available. However, dithering comes at the price of slower redraw times, because the computer must calculate how to diffuse the color errors as it is redrawing the image. When any new window is created, **IPLab** initially displays it without dithering. When you are operating your monitor in 24-bit mode, setting the dither on or off has no effect.

5.2.2.8 Display-Independent Processing

It is important to realize that **IPLab** keeps your image data values intact, regardless of dithering and regardless of your monitor's bit depth. Changing the characteristics of your display does not change the values of your data. This ensures the accuracy of your image processing computations no matter how you display your data or how many colors your monitor is displaying.

5.3 Copy-Paste Operations

There are five kinds of paste buffers in **IPLab**:

- **Data**
- **QGraph/Perspective**
- **Script**
- **Segment overlay**
- **Drawing objects overlay**

Each of these paste buffers is separately maintained, so you cannot, for example, copy something from an overlay and paste it onto data. **IPLab** manages the different paste buffers “behind the scenes” for you when you choose the appropriate copy and paste commands.

Data: Data is always saved in the data paste buffer as raw data (binary pixel values) with the same data type as the data window from which you copied it, and never as text. This way, you can paste data into other windows of the same type.

Non-byte data must be handled in a special way when you quit **IPLab** or when you switch to another application under Finder, since most other programs cannot understand non-byte data. In this case, the display-normalized data gets copied to the Macintosh Clipboard in such a way that its contents may be pasted into other programs. That means that all image displays can be copied into the Macintosh Clipboard and pasted within other programs.

QGraph/Perspective: When you are viewing an image as QGraph or perspective and you perform a **Copy** operation, the view you see on the screen in copied to the Macintosh Clipboard and can be copied into the Scrapbook and documents in other programs. The data paste buffer is not affected.

Other: You use the script paste buffer for copying and pasting between scripts, and the Overlay Paste Buffer for copying and pasting between overlays. The contents of the Script Paste Buffer, the Segment Paste Buffer and the Draw Object Paste Buffer are only used internally by **IPLab**. These paste buffers are not available for pasting data into other programs.
6 Scripting

6.1 What is a Script?

An *IPLab* script is like a flexible “macro-maker” tool. It is a recording of a sequence of commands. It is effectively a program that you create as if you were applying menu operations interactively to your image data. When you run a script, the scripted commands are executed in the order you placed them. *IPLab* scripting allows you to customize the software to your own needs, by grouping together commands that describe your protocols and algorithms. Scripts may be saved and run with a few keystrokes, creating powerful new “commands” which are customized to your specific application.

A Script Next to the **Script Commands** Palette

Script editing is done using two windows: the script window and the **Script Commands** palette. The script window contains the recorded commands. The **Script Commands** palette contains buttons for editing and controlling scripts.
6.2 How to Script

Make an *IPLab* script in three or four steps:

1. Create new scripts by choosing the **New Script** command from the **Script** menu. You can also use the **Open Script** command to read an existing script file from disk.

2. Record commands for your script by using *IPLab* normally:
   - Picking commands from the menus
   - Clicking buttons on the **Script Commands** palette
   - Pressing key combinations (like ⌘-O for Open)

As you record commands, you will see the command names appear in the command list on the left side of the script window. New commands will appear above the first highlighted command, or at the top of the script if there are no highlighted commands.

There are a few interactive commands that cannot be recorded in a script. These commands will be dimmed when you record your script.

3. Set up the command parameters for each command, just as you would normally do. When you record a command, its dialog box (if it has one) appears. Set its parameters and click **OK**. The script saves the parameters you set.

   You can alter recorded settings by double-clicking on a command in the command list. The dialog box will open again, letting you change the parameters.

4. You have the option of adding comments to your script. Comments can explain the purpose of the recorded commands, making it easier to use and edit the script.

After that, save your script by clicking **Save** or **Save As** on the **Script Commands** palette. When you run your script, the command list's commands will run in order from top to bottom.

The rest of this Scripting chapter tells you how to use each of *IPLab*'s scripting tools. If you like, you can skip around this chapter to learn only the details you need right now.

6.3 Using the Script Window Buttons

The script window has several buttons on its top edge. Below that, it contains the scrollable list of commands and associated comments.

![Features of the Script Window](image)

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ON</td>
<td>Turns the script on and off</td>
</tr>
<tr>
<td>Continue</td>
<td>Continues running the script when suspended</td>
</tr>
<tr>
<td>Run</td>
<td>Runs the script</td>
</tr>
<tr>
<td>Display info</td>
<td>Displays script info and settings</td>
</tr>
<tr>
<td>Docks</td>
<td>Docks the Script Commands Palette</td>
</tr>
</tbody>
</table>

136 Scripting
6.3.1 Record ON/OFF

This button turns script recording on and off. Normally, when a script is the front window, it records every menu command you select. You may want to execute a menu command without the script recording it.

Turn script recording off by clicking the **Record ON** button in the script widow. This button will change to **Record OFF**, and the **Script Commands** buttons will dim. You cannot open the dialogs for recorded commands when recording is turned off. Menu commands will now work normally, instead of being recorded in the script. You can still run script when recording is turned off.

Click **Record OFF** to turn script recording on again.

6.3.2 Script Info Button (i button)

Clicking the script info button opens up a dialog containing a description field and preferences for the script:

![Script Info Dialog Box]

Enter a summary of the script in the **Description** field. You can use this field to tell users what operating conditions the script has. For example, the script may only work with color images.

Use the two sets of radio buttons to control the script's behavior. The **After a Cancel** options determine if you'll see an error message when something halts the script before it finishes. The **New Images** options determines if the script will hide new images to increase its speed. (Displaying image data can slow down a script, particularly on older computers.) Please see the **Show/Hide Image** command on page 198.

The information on the dialog's right side tells you how full the script is, which helps you when you write large scripts. It also tells you the version of IPLab that wrote the script.

6.3.3 Continue and Run Buttons

Click the **Run** button to start the script at its first command.

Click on the **Continue** button to start the script at the first highlighted command. That lets you run only part of the script, which is particularly useful when your script or experiment was interrupted, or when you are testing a new script.

IPLab will immediately hide the script while it executes the script's commands.
6.3.4 Script Docking button (l button)

The script docking button is the vertical bar on the script window's right side. It docks the Script Commands palette by moving the palette to be right next to the script window.

6.4 Recording the Script

Recording commands adds them to the command list. You can make some commands interactive, so their dialog boxes appear for the user. You can also add comments to explain your script.

6.4.1 Command List

The command list is the sequence of recorded commands listed on the script window's left side. Every command recorded by the script gets added to the command list. That includes commands picked from the menus, buttons clicked on the Script Commands palette, and key combinations (like \texttt{z-O} for Open).

To record a command, use it as you would normally:
- Pick the command from a menu
- Click a button on the Script Commands palette
- Use a key combination (like \texttt{z-O} for Open)

You will see the scripted command appear on the left side of the script window. Recording commands adds them above the first highlighted line. If you want to add the command to the bottom of the script, select the END command; the new command will be added above that. If no command lines are highlighted, then the new command will be added to the top of the script.

To highlight a command in a script, simply click on it. To highlight several commands, click and drag the mouse or use the Shift key to extend the highlighted selection. Because you control the new command's insertion point, you do not have to record the entire script in order, from top to bottom, which would require much planning. Instead, you can always return to an earlier point in the script and add or change commands.

A few commands are meant to be interactive, and thus cannot be scripted (Preferences, for example). These commands will be grayed out (unavailable) when a script is open with recording turned on. You can temporarily toggle the button to Record OFF to use these commands.

You will notice that every script ends with the END command. Highlight the END line to record commands immediately above it. When a running script reaches the END command, the script returns control of IPLab to you.

Note: "Recording a command in a script" and "scripting a command" mean the same thing.
You can change a command's parameters after the command is scripted. Double-click on the scripted command's name. Its dialog box will open again, and you can change its settings. You can also select the command and then click on the **Dialog** button on the **Script Commands** palette.

### 6.4.2 Interactivity: Making Dialogs Open

You can make scripted commands interactive, so that their dialog boxes, if any, appear when the user runs the script.

If the command can be made interactive it will have a blue plus or minus symbol next to it. Click on the symbol to toggle that command from non-interactive to interactive, or vice versa.

- The blue minus sign, −, means that the command is not yet interactive. When the script runs, it will use the saved parameters for this command.

  Click on the minus symbol to change it to a plus, making the command interactive.

+ The blue plus sign, +, means that the command is interactive. When the script runs, this command's dialog box will open. The user can then set the command's parameters. The script will continue running when the user clicks **OK** in the dialog box.

  When the interactive command's dialog opens, it will contain the parameters you entered when you recorded the script.

  Clicking **Cancel** on an interactive command's dialog suspends (halts) the script. The user can restart the script on the same line by clicking **Continue** on the script window or on the **Script Status** palette.

In the example script shown on page 138, the **Full Acquire** command is not interactive, while the **Segmentation** command is interactive. When this script runs, the **Full Acquire** command will use the bin size and other parameters that were recorded with the script. However, the dialog box for the **Segmentation** command will open. The user can adjust the **Segmentation** minimum and maximum threshold values to segment the right parts of the image. When the user clicks **OK**, the script will continue with the next command by quantifying the segments.

### 6.4.3 Commenting Your Script

You can enter a comment for every command in the script. Comments make editing scripts easier, and they help users understand exactly what scripts do. Comments can identify portions of the script, explain the reasoning behind a group of commands, and reveal the settings for a command so that the reader does not have to open the command.

To enter a comment, double-click in the comment column to the right of the command. You can also select the command and click the **Comment** button on the **Script Commands** palette. Type the comment and click **OK**.
6.5 Using the Script Commands Palette

This section describes the buttons on the Script Commands palette:

![Script Commands Palette](image)

### 6.5.1 Editing the Script:
Cut, Copy, Paste, Clear, Duplicate

You must use the appropriate buttons on the Script Commands palette to Cut, Copy, Paste, Clear, or Duplicate commands that are highlighted within a script. If you try to use the Edit menu commands, the script will record the Edit menu commands as if they were meant to edit image data.

Cut and Copy places the highlighted commands, along with their parameters and comments, into a special paste buffer for scripts. Cut also clears the highlighted lines from the script. You can then paste those commands elsewhere in the script or into a different script.

### 6.5.2 Saving and Printing the Script:
Save, Save As, Print

To save or print the script, you must use the Save, Save As, and Print buttons on the Script Commands palette. Using the File menu commands would simply record those commands in the script.

**Note:** This means that your scripts can save your image data for you when the Save or Save As commands are recorded in the script.

### 6.5.3 Insert

The Insert button pastes the contents of another script into the current script. To insert a script, select the line above which you want the other script inserted. Then click the Insert button. An Open Script dialog appears; browse to and choose the script you want to insert. IPLab will then copy those commands to your current script.

### 6.5.4 Revert

When you click the Revert command, the script will change back to the version stored on disk. You will lose all changes you have made since you last saved.

### 6.5.5 Dialog, Comment

The Dialog button opens the dialog box (if any) for the selected command. The Comment button opens the dialog for editing the selected command's comment. You can also double-click on the command or on its comment.
6.5.6 Alert

Clicking this button adds the Alert command to the script and opens its dialog box (below, left). When the script runs the Alert command, it will display a simple message box with an OK button:

Alert Dialog Box  The Resulting Alert Message

The Alert dialog box prompts you for the text you want to appear in the alert.

If you check the box labeled Close Alert After _____ Seconds, the alert will go away after the specified number of seconds, even if the user does not click on OK. If you check this box and enter 0 seconds, the alert will never appear; only a system beep will sound.

You can also control what IPLab will do after the alert closes:
- Continue executing the script.
- Suspend the script and bring the image to the front so that the user may operate on the image before returning to the script.
- Stop execution of this and any other sub-scripts that it calls.
6.5.7 Query

Clicking this button adds the **Query** command and opens its dialog box (below, left). **Query** lets the user of your script control what the script does by letting the script see which button the user clicks on. The **Query** command prompts the user with a question and stores the number of the button clicked.

![Query Dialog Box](image)

**Query** Dialog Box

Type the **Title** and the text of the question in the two text-entry boxes. Then enter the index of the **IPLab** variable you want to hold the results. It is a good idea to spell out the choice(s) you want the user to make.

If you want to use three buttons instead of just two, check the **Use Button #2** option.

To edit the text for the three buttons, click on the buttons numbered #0, #1, and #2 to bring up another dialog:

![Dialog Box for Query Button Text](image)

**Dialog Box for Query Button Text**

When the script executes the **Query** command, the user sees this dialog:

![Resulting Query Prompt](image)

**Resulting Query Prompt**
The user will click on one of the three buttons, and the number of that button (0, 1, or 2) will be stored in the variable. Your script would use one or more If commands to jump to various parts of the script, depending on the variable's value. In the example Query shown above, the script would jump to a section that acquired color images, a section that acquired grayscale images, or to the end of the script (thus quitting the script).

6.5.8 Enter Var (Enter Variables)

Clicking this button adds the Enter Variables command to the script and opens its dialog box. The Enter Variables command prompts the user to enter numeric values, which will be stored in variables. Other IPLab commands can then use these values.

![Enter Variables Dialog Box While Recording the Script](image)

Fill in the Title and Description fields to tell the user what they should do, or for what sort of information they are being asked. In the Prompt field, tell the user what specific numeric information they should give. The user's answer will be stored in the variable named in the Variable To Use field. You can specify any variable from variable #0 to #511.

If you like, you can display a default value next to a prompt. If you check the Display Constant checkbox, then the number in the Display Constant field will be shown as the default value for that prompt.

If you do not use the Display Constant value, then the prompt's value will default to the value stored in its variable.
When the script executes the **Enter Variables** command, it prompts the user with this dialog:

![Resulting Enter Variables Dialog Box]

In the example dialog box shown above, the values for the first three prompts (exposure time and the two bin sizes) are the numbers currently stored in the variables for those prompts (variables #255, #11, and #12). Those values might be different the next time the user runs the script. However, the value for the last prompt (number of frames) is the default number entered in the **Display Constant** field. This is a good way of reminding the user that, in this example, five is a good number of frames.

### 6.5.9 Label

This button records the **Label** command in the script and opens its dialog. The **Label** command marks a position in the script, which is then referred to by the **If**, **Jump**, and **Loop** commands. You can think of each scripted **Label** command as a landmark that **If**, **Jump**, and **Loop** use to navigate around the script. You must place your labels before you can record an **If**, **Jump**, or **Loop** command.

![Label Dialog Box]

Each **Label Name** must be unique within the script. (These landmarks would be useless if they looked just like other landmarks.)

Once recorded, the **Label Name** appears in the command list as underlined text.

You can also use labels to mark sections of your script for your own uses.
6.5.10 If

Clicking this button records the If command in the script and opens its dialog. The If command provides a conditional jump control structure. If the logical relationship is true, then control of the script will jump to the specified label. If the relationship is not true, then the script will continue with the next command following the If command.

At least one Label command must have been recorded before you can script the If command.

![If Dialog Box]

In the If... section, set up your logical relationship (for example, "variable #10 = the number 1"). You can compare the values of two variables, two numbers, or a variable and a value.

In the Then... section, choose the target label. The script will continue running from this label if the relationship in the If section is true. Otherwise, the next command in the script will be executed.

6.5.11 Jump

This button adds the Jump command to the script, and opens its dialog as well. The Jump command unconditionally moves control of the script to the specified label. When the script executes the Jump command, the script will continue running from the named label. This means that your scripts do not have to be written linearly in the order of execution from top to bottom.

At least one Label command must have been recorded before you can script the Jump command.

![Jump Dialog Box]

The Jump command can jump to any existing label.
6.5.12  Loop

Clicking this button adds the Loop command to the script, and opens its dialog. The Loop command provides a simple backward looping capability: the loop will perform all commands between a label and the Loop command for the given number of iterations.

To create the loop:
1. Record the Label command.
2. Record all of the commands you want to repeat.
3. Record the Loop command, which will return control of the script to the label's position in the script.

At least one Label command must have been recorded before you can script the Loop command.

The Loop Dialog Box

First, choose the label at the start of your loop.

Then set the number of iterations. Because the Loop command comes last in this control structure, the looped commands will always run at least once. The Loop command's iteration count always starts at 1.

The Current Count field contains the current iteration. If something (such as an Alert set to stop scripts) interrupts your script, you can finish running the script without restarting by clicking on the Continue button. If you wish to reset the iteration count, you must open the dialog for each the Loop command that you wish to restart, and click on Reset.

The # of Iterations parameter can be read from a variable by toggling the # symbol to the VAR symbol. This means that your loop can set its own number of iterations.

The iteration count that you see in the dialog of a Loop command is not available to any other command in the script. If you want to use the iteration count in another command, you should increment an IPLab variable by one every time the loop runs. You can then use that variable in the other commands.

6.5.13  Delay

Clicking this button adds the Delay command to the script, and opens its dialog. The Delay command helps time your script. It pauses the running of the script, either for a set duration or for a total amount of elapsed time.

While the script does nothing, the cursor changes to a hollow white arrow, N, to indicate that the script is paused.

The actual delay time is accurate to within several milliseconds, depending on the computer's speed.

You can force the script to cut short the delays created by the Delay or Wait for Timer options. Simply press a keyboard key other than Escape while the script is being delayed. The script will instantly run the next command. If you press Escape during a delay, the script will stop running, just as it normally would.
The **Delay** function pauses the running of the script for a set duration. Click the **Delay** radio button and enter the amount of time for which the script will do nothing.

The rest of the **Delay** dialog, the "timer" portion, lets you measure the passage of time and delay the script until a certain amount of time has passed.

- **Using Timer #...:** Each script has eight storage spaces for remembering times: they are timers #1-8. Choose the timer in which you want the time to be stored, or in which the time has already been stored.
- **Start Timer:** Choose this radio button and click **OK** to record the **Delay: Start Timer** function. When the script runs this line, it will record the current time in the chosen Timer.
- **Wait for Timer...:** Choose this radio button and click **OK** to record the **Delay: Wait for Timer Completion** function. This function forces every iteration of a loop to occupy the same amount of time. When it runs, the script stops and does nothing until the set duration has elapsed since the time was recorded by **Start Timer** in the chosen Timer.
- **Beep if Overdue:** If the script reaches the **Delay: Wait for Timer** line after the wait time is over, then it will beep (actually, it will make the system alert sound).
- **Store Elapsed Sec...:** Choose this radio button and click **OK** to record the **Delay: Store Elapsed Seconds since Start or Wait** function. When the script runs this line, it stores the amount of time that has passed since either:
  - the last **Delay: Start Timer** command, or
  - the end of the last **Wait for Timer**

It stores this elapsed time in a variable of your choice.
6.6 Running, Continuing, and Stopping a Script

Besides using the Run and Continue buttons, you may also run a script with the Run Script command from the Script menu. That works even when the script is not the active window. This manual describes the Run Script command on page 290. IPLab executes the script's commands without opening a script window. If an error stops the script, the script window will be opened with the offending line highlighted. If an alert stops the script, or if you press Escape or Command-period to stop the script, the next command to be executed will be highlighted so that you may press the Continue button.

After you start a script running, it continues to run until one of three things happen:

- It reaches the END command
- You stop the script by hitting the Command-Period or Escape keys
- An error halts the script

When the script stops running, the script window is opened so that you may quickly edit it or run it again.

If you stop a script by hitting the Command-period or Escape keys, the script finishes executing the current command and then stops, opens the script window, and highlights the next command in the script. At this point you may activate other windows and use any commands you like. Use the Continue button (or the Script menu's Continue command) to continue running from the highlighted command. This feature is particularly useful in conjunction with a Delay command (which is described on page 146).

When a script stops due to an error, it opens the script window and highlights the command that generated the error. It also opens the yellow Script Status palette:

![Script Status Palette](image)

You can cancel the execution of the script and its sub-scripts by clicking on the Cancel All button in the Script Status palette. If the error can be rectified by, for example, opening or renaming an image window, you can do so and then click on the Continue button.
6.7 Running One Script from Another Script

You can record the Script menu's Run Script command within a script. When the main script runs that command, it will open and run the second script. The script activated by the Run Script command acts as a king of "subroutine" to the main script.

When you open the dialog for the scripted Run Script command, you will see, at the bottom, the name of the script to be run:

There are certain limitations which you should be aware of before making extensive use of this feature.

Call Depth: You may nest up to four Run Script commands at one time. That is, script A may use Run Script to run script B, which may run script C, which may run D, which may run E.

Cycles: IPLab does not prevent you from using Run Script on the same script twice during one execution (for example, script A runs B which then runs A), but it is not recommended. You must be very careful to prevent the calls from exceeding the call depth. Also, any loop counts in each script are re-initialized on each new call.

Continue: If an embedded script stops for any reason (for example: error, Alert with the Stop All Scripts option, Command-Period), then the Continue button will continue that script, but not any script which may have called it.

Portability: If the script being run is located in the "IPLab Scripts" folder, then you can transfer the script to another computer (still in the "IPLab Scripts" folder), and the scripts will run. Otherwise, your main script might not know where to find the script being run.

Memory and Speed: Opening scripts and other files can slow the running of your scripts. For example, Suppose script A runs script B. If B is not in memory (that is, already open in a window) when script A calls it, script B must be read from disk and executed, and then closed. The time it takes to open and read script B slows the execution of script A. This is especially true if script A calls script B from inside a loop. If script B is already open when A is run, then IPLab executes script B from memory, and no file operations take place.
6.8 Error Handling

If an error occurs while a script is running, the dialog for the command being executed may appear. A variety of conditions might generate an error that brings up a command’s dialog. One of the most common reasons for this type of error is that you have entered the name of a file or window that does not exist at the time the script reaches the command.

When the command's dialog box appears, you may be able to continue running the script by entering an appropriate value and clicking **Continue**. Note that the script will not remember this new value. The parameters associated with the command are not changed in the script. If the command is executed several times within a loop, the same error stops the script each time.

You may make use of this feature as an opportunity for interaction with your script. For example, you may want to display a sequence of images in a slide show, but you do not wish to specify the order in which to display the images beforehand. To do this, script a loop around an **Open: from File List** command. Create the file list using names of files that do not exist. Each time the script encounters the name of a nonexistent file, the **Open** dialog will reappear to let you select a desired file.

You can permanently change the command parameters in the script at the point of the error. You can stop the script by selecting **Cancel All** in the **Script Status** palette. At this point, the script will have reappeared with the error-generating command highlighted. Double-click on the highlighted command, to see the command's dialog, if it has one. You may then change the command parameters, close the dialog, and click on **Continue** to continue the script.

6.9 Startup Script

If you have a script named "Startup Script" in the "IPLab Scripts" folder, then that script will be run every time **IPLab** starts. The Startup Script is a good place to put hardware initialization commands. Also, if you know you will always do the same operations when you start **IPLab**, you can place those commands into the Startup Script. If you want to discontinue your Startup Script, simply remove the Startup Script file from the "IPLab Scripts" folder.

6.10 IPLab Scripts Folder

The "IPLab Scripts" folder is inside "IPLab 3.6 User Folder".

IPLab 3.6 Folder: IPLab 3.6 User Folder: IPLab Scripts

Besides being a good place to store your scripts, the "IPLab Scripts" folder is useful when scripting. Scripts that use the **Run Script** command to execute other scripts are portable when all of the scripts involved have been saved in the "IPLab Scripts" folder. You can move the scripts to the "IPLab Scripts" folder on another computer, and the main script will still be able to find its subroutine scripts.
7 Menu Reference

7.1 Introduction

7.1.1 Overview of the Menus

This chapter provides the details of operation for each command in the IPLab menus. The commands are organized according to their menu. Read the descriptions in the Menu Reference chapter to learn how to use each command. If you are not certain which command you need, look for the menu that describes your basic task, and the command you need should be within.

The menus are organized into groups of commands with similar functions.

- The File menu contains commands that manage and print files.
- The Edit menu contains basic commands for selecting and altering the information content of an image and overlay. The View menu allows you to alter the way the image data in the active window is seen without changing the image.
- The Enhance menu provides commands that improve the appearance of the image.
- The Analyze menu lets you 1) select regions of the image to measure, 2) choose measurements to perform, and then 3) perform the measurements.
- The Math menu provides advanced commands that mathematically alter the values in a data window.
- The 3D menu provides commands for visualizing and processing three-dimensional data. This menu appears only when the 3D extension (included with IPLab) is installed.
- The Camera menu appears when you install an optional camera control, enabling IPLab to operate a camera system.
- The Control menu appears when you install an optional, hardware-controlling extension. These commands let you control motorized stages, filter wheels, shutters, and other hardware on microscopes and other systems.
- The Ext. (Extensions) menu is displayed when you install optional processing extensions, such as our Ratio Plus package. Processing extensions give IPLab additional functionality.
- The Script menu contains all of the variable-setting and scripting commands, which let you make time-saving recordings of commands that are similar to macros. At the bottom of the Script menu is a list of the open scripts.
- The Window menu contains commands that help you organize the windows on the screen. At the bottom of the Window menu is a list of the open windows.

Camera controls and all extensions (3D, control, and processing extensions) must be installed in the same directory as, and at the same level as, the IPLab program. Please see the Extensions manual for information about these additions to IPLab. You can use the commands provided by extensions just as you would any other command. Extensions may be supplied by Scanalytics or by third party developers, or you may write them. Detailed instructions for how to write your own extensions are contained in the Writing Extensions chapter on page 303 of this manual.

7.1.2 Default Parameter Settings

When you invoke most IPLab commands which have an accompanying dialog, the dialog will by default appear with the settings you last used. This feature allows you to pay attention to only those parts of the dialog which you wish to change, permitting you to perform the function more quickly. The few exceptions to this rule are noted in the descriptions of the commands.

Scripting also makes an exception to this rule. When recording a command into a script, the scripted settings do not affect the default settings that will appear during interactive use of the command.
7.1.3 Interactive vs. Script Operation

When you create or open a script window, almost all commands that you then select will be recorded in the script but will not actually be executed. For example, if you record the **New Image** command in a script, the command will appear in the script window but a new window will not be made. This lets you edit the script until you are ready to run it and make the new image.

Some commands have different dialog boxes when they are run interactively than when they are run while a script window is active. These minor alterations handle the differences between running the command live versus running the command in an automated (scripted) fashion. If a dialog is different when scripted, the command’s description will make note of it.

As noted in the previous section, the settings of a scripted command will not affect the default settings of the same command when it is used interactively.

7.1.4 Errors

Errors generally produce beeps or alerts which describe the error situation. However, if an error occurs during the execution of a command while a script is running, the dialog for the command which was executing may also appear. At this point you can stop the script by selecting **Cancel** in the dialog. Sometimes, you may be able to continue the script execution by entering an appropriate response in the dialog. If you select **Cancel** from the command’s dialog, or if the error does not cause a dialog to appear, then the script which is running reappears with the error-generating command highlighted. At this point, you can double-click on the command name to see the dialog for this command. Once the error situation is fixed (usually by correcting an inappropriate setting), clicking **Continue** will set the script going from the selected command.

7.1.5 Commands Which Operate on Direct Color Images

Most commands operate on any data type. A few commands cannot be applied to the direct color types, color 24 and color 48. If you try an operation on a direct type image and get an alert which tells you to “Use Split Color Channels First,” you must first use the **Split Color Channels** command (Math menu) to separate the color image into its color components. Then apply the desired processing to the separate color coordinate windows. After processing each frame, you may use the **Merge Color Channels** command (Math menu) to put the color image back together.

7.1.6 Analyzing Regions

**IPLab** provides several optional methods for you to get measurement and statistical information about regions of the image. These commands are found in the **Analyze** menu. Before performing the measurements, you must use the **Set Measurements** command to tell **IPLab** which measurements you wish to perform. You may also need to use the **Measurement Options** command to specify what segment color you are measuring and the name of the results window.

You may then use an ROI selection tool to outline a region, and choose the **Measure ROI** command to perform the measurements. If this is the first set of measurements you have performed, **IPLab** creates a new window behind all of the other windows and stores the results there. You may continue to perform measurements on other regions within the same image or other images. Each new measurement goes into the previously created results window. When you are finished performing measurements, you can easily bring the results window to the front by finding it in the list of open windows under the **Window** menu.

**IPLab** also lets you measure several regions at once using the **Measure Segments** command and the **Quantify Segments** command. Segments can be created using the **Segmentation** command, or you may paint **Segments** by hand using the segment drawing tools found on the **Tools** palette.
7.1.7 Apple Events and AppleScript

The following information should tell you how to control IPLab from outside the program, including from AppleScripts. (Apple and AppleScript are trademarks of Apple Computer, Inc., registered in the U.S. and other countries.)

IPLab for Macintosh accepts the required Apple events (Open Document, Print Document, Quit Application) as well as DoScript. This means that another application can tell IPLab to open a specific image and then run an IPLab script which operates on that image. IPLab becomes the front application whenever it receives an Apple event. Since scripts may take some time to execute, the other application should ignore time-outs.

IPLab for Macintosh cannot operate as a background task.

The following information is for programmers who wish to send the DoScript event from a separate application to IPLab. It is taken from the Apple Events Registry, published by Apple Computer and sold through APDA. For more information, see that document.

Event Class: kAEMiscStandards (‘misc’)
Event ID: kAEDoScript (‘dosc’)
Direct Parameter: This parameter tells IPLab where to find the script file.
   keyDirectObject: Descriptor Type: IPLab only supports typeAlias (‘alis’)
Reply Parameters: The absence of a keyErrorNumber parameter in the reply indicates that the event was handled successfully. IPLab returns 1 if any error occurred while running the script.
   keyErrorNumber: Descriptor Type: typeLongInteger

One way to send Apple Events to IPLab for Macintosh is through AppleScript. Below is a very simple AppleScript, which tells IPLab for Macintosh to start, open a file, and then run a specific IPLab script.

```
An AppleScript Example
```
This script can actually be recorded with Apple’s *Script Editor*. The procedure is as follows:

1. Start *IPLab for Macintosh*.
2. Return to the *Finder* and start Apple’s *Script Editor*.
3. From the Controls menu in *Script Editor*, select the command *Record*.
4. Switch to the *IPLab for Macintosh* application using the pop-up list of applications in the upper right corner of the computer screen.
5. In *IPLab for Macintosh*, open the *Chromosomes* image.
6. Select the *Run Script* command from the *Script* menu. Find and open the script named *Subtract Background*.
7. Answer all of the questions, or OK the alerts that the script presents you.
8. Quit *IPLab for Macintosh*.
9. You will be returned to the *Script Editor*. From the Controls menu, select Stop. The AppleScript is written for you.

### 7.1.8 Apple Menu

#### 7.1.8.1 About IPLab

This command shows the startup window for *IPLab* and provides copyright and version information. It also displays a list of extensions and control modules that were loaded into *IPLab* as the program started. The *About IPLab* command is not scriptable.

#### 7.1.8.2 Apple Menu Items

The *About IPLab* command is followed by the list of your Apple menu items. Apple menu items are not scriptable. They can be accessed regardless of the active window type.
7.2   File Menu

7.2.1   New Image

Use this command to create a new, empty data window. Being empty of data, all of the pixels will have value 0. All of the grayscale images will be made with the monochrome CLUT.

A dialog will prompt you for the data type and size of the data window you wish to create. You may also create an empty image sequence of more than one frame.

![New Image Dialog Box](image)

**New Image** Dialog Box

**Name:** This will be the name shown in the title bar of the new image.

**Dimensions:** Use these radio buttons (described immediately below) to dictate the size of the new image. The size of the image (width x height) is measured in pixels.

- **Size of Clipboard Image:** The new image will be the size of the last selection you copied to the clipboard when you select this radio button. This is very helpful when you have copied part of an image and want to paste it into a new image with the right size.

- **Size of Front Window ROI:** The new image will be the same size as the selection in the front-most, or active, window.

- **Custom Size:** You can specify the exact width and height, in pixels, of the new image.

**Frames:** You can make your new image as a single frame or as a multiple-frame sequence. Please see the Image Window section on page 9 for a description of sequences.
Data Type: Pick the data type based on what type of data you want to put into the new image. Your choices (Byte, Short Integer, Unsigned16, Long Integer, Floating Point, Color 24, and Color 48) are described in the Operation chapter on page 129.

Very simply, use Byte type for grayscale video data, Color 24 type for color video data, and Short Int or Unsigned 16 data type for grayscale digital data. Use Floating Pt. data for storing the results of calculations.

7.2.2 Open

The **Open** dialog box lets you display images that already exist and are stored on your hard drives or disks. The current directory is shown in a pop-up menu at the top-left corner of the dialog box. The images in that directory are named in the list box below the directory bar. The list only displays files that are one of the standards types recognized by **IPLab** (IPLab, PICT, TIFF, FITS, and Text).

When you open a file, **IPLab** creates a new active window labeled with that filename and displays the image in the new window. If a CLUT is attached to the file, **IPLab** attaches that CLUT to the image. If no CLUT is available from the file (which is the case for those PICTs and TIFFs that contain Direct Color images), **IPLab** attaches the System CLUT to the image.

7.2.2.1 Opening a Single File

Simply put, you just need to find the file to be open and click the **Open** button. The **Single File** radio button, which you use for the everyday opening of images, will become selected automatically as soon as you click on a file in the list box.

To find the file, navigate your computer’s folders (directories) using the **Desktop** button, the **Open** button, and the directory pull-down bar (which says **IPLab Tutorial Images** in the example, above). When you click on the **Desktop** button, the file list will show the folders and images on your computer’s desktop. To see what’s inside any folder, double-click on it or select it and click the **Open** button. You can also move up in the folder hierarchy by selecting a higher folder from the directory pull-down bar.

When you find the file you want to open, double-click its name in the list or select it and click the **Open** button.

When you highlight one of the files in the list by clicking on it, information about the image is displayed on the right side of the dialog box. A preview of the image may be displayed. The file size (in kilobytes) and the file type of the image will be displayed in the **Size** and **Format** fields. In addition, the amount of space left on the hard drive or disk is displayed under the **Open** button (in the example dialog, 185.5 MB are available).
To see a preview of an image, click the Show Preview checkbox on the right side of this dialog box. The image must have already been saved with a preview image. You can do this in IPLab by using the Save As command with the Save Preview option checked.

7.2.2.2 Opening Images from File List

Instead of finding the file you want to open from the scrolling list of names, you can open the next file named in a file list. A file list is a collection of the names of image files that you put together. This is a good way of specifying ahead of time which images are to be opened and in what order. To use this option, click the From File List radio button at the bottom of the Open dialog, and then click the Open button.

The From File List Option at the Bottom of the Open Dialog Box

File lists are made with the Edit File List command (File menu). The file list to be used is picked with the Select File List command. When you click the From File List radio button, the selected file list will be displayed. (In the part of the dialog box shown above, the file list being used is named "MPG List 1").

The file to be opened, when you first use this option, is the file that was highlighted in the Edit File List dialog box when you closed it. Thus, you can set which file in the list will be opened first. After that file is opened, the pointer in the file list will be incremented so that the next time you (or a script) use Open: from File List, the next file on the list will be opened.

Please see the descriptions of the Edit File List and Select File List commands on pages 169 and 171.

7.2.2.3 Opening Indexed Files

If your images are named with a base name followed by an index number, instead of opening a file by using the scrolling list of names, you can open the next in a group of sequentially numbered files. To do this, your files must share the same name and be sequentially numbered (e.g. Control0000, Control0001, Control0002, etc.). If you open Control0000 first, the next indexed file you open would be Control0001. This is a good way of controlling which images are opened and in what order. To use this option, click the Indexed File radio button at the bottom of the Open dialog, and then click the Open button.

The Indexed File Option at the Bottom of the Open Dialog Box

In order to have indexed files, you can either name the files yourself with a base name and an index number, or you can use the Save As command with the Indexed File option.

In order to open indexed files, you must use the Set Indexed Name: for Open command (File menu) to tell IPLab what base name and index number to use. Because you can specify the index number, you can start by opening any image in the sequence. (In the part of the dialog box shown above, the base name Control and the index number 0 was set.)
When the **Indexed File** option is used, and the **Open** button is clicked, the displayed file (Control0000, in our example) will be opened. The index will then be incremented so that the next time you (or a script) uses **Open: Indexed File**, the next indexed file will be opened.

Please see the description of the **Set Indexed Name** command on page 172.

### 7.2.2.4 About Opening Specific File Types

**IPLab** reads specially prepared text files. For the exact format of **IPLab** text files, see page 123.

**IPLab** reads uncompressed TIFFs whether they are single frames or sequences of frames.

**IPLab** also reads the FITS (Flexible Image Transport System) images.

If you have QuickTime in your operating system folder, **IPLab** can open compressed PICT files. If you do not have QuickTime, and the file you try to open is a compressed PICT, or if your QuickTime does not have the appropriate decompression routines, you will get a message telling you that you need certain QuickTime compression software. Only Apple Computer provides the QuickTime compression software; they provide it as part of their operating system.

### 7.2.2.5 Mounting Multiple Volumes

In order to support batch file processing through the file list and through the use of filenames in scripts, **IPLab** makes use of the volume names of the disks you mount on your system. If you mount multiple disks with the same volume name by inserting them into any of the removable disk drives, **IPLab** may have difficulty opening any files on those disks. It is best to make sure that each of your disks has unique volume names before entering files onto the file list. You can change the name of any mounted disk volume by highlighting its name in the Finder and typing in the new name. Your scripts and file lists may also have trouble finding files if you change the name of your hard drive or file servers where images are located. If you need to undo changes to your hard drive’s name, you may need to first turn off file sharing.

### 7.2.2.6 The Open Command in Scripts

There are several items to note when you select the **Open** command while scripting.

1. If you select either **Open: from File List** or **Open: Indexed File** while making a script, the term `<scripted>` appears where the name of the file list or indexed file would normally appear. This is because the name of the file list or indexed file will be determined when the script is run.

2. If your script has an **Open** command in it, you can bring up the dialog of that command to see what file you have selected. If you chose to open a single file, the filename you have selected will appear next to the radio button labeled a **Single File**.

3. **Important:** Suppose you choose the **Open** command, select the file **Washington**, and click **Open** to place the command in a script. You then go back and double-click on the **Open** command in the script to re-display its dialog. The highlighted filename in the dialog’s scrollable file list may not match the one you had selected, as is shown in the figure below. If you wish to accept the filename you have already selected (“Washington”), you must be sure to click on the **Cancel** button in the dialog. If you click on the **Open** button at this point, the highlighted file in the dialog’s scrollable file list (“16Bit XRay”) will be selected instead.
Open Dialog Box, as Observed from a Script.
Notice that the name of the file to be opened is different than the name of the file highlighted in the list.

7.2.3 Open As

This command uses almost exactly the same dialog as the Open command. However, it allows you to open any file and treat it as if it were a file of any other type. So, for example, you can open files that are internally labeled as Text files, but treat them as if they were TIFF files. This is very handy when bringing images files to the Macintosh from other computers.

Open As Dialog Box

When you bring a file to the Macintosh from some other computer system, the file must be translated into a “Macintosh file”. One of the features of a Macintosh file is a hidden attribute called the file type. The file type takes the place of filename extensions which are often used on other computers. There are various utilities for translating foreign files into Macintosh files, and most of those utilities assume that the files you are transferring are TEXT type even though they may be TIFF or raw binary data.

In the lower right side of the Open As dialog, click on the pop-up list of file types to select the type you want IPLab to treat this file as. This type overrides the hidden attribute that is attached to the Macintosh file. When you click the Open button, IPLab will make a new window with the image in it.
Remember that the **Open** command uses the hidden file type attribute to open a file, while the **Open As** command lets you override the file type attribute and open the file as if it were any type you want.

### 7.2.3.1 Reading Foreign Files

*IPLab* also allows you to read foreign data files which you may obtain from a variety of sources. Select the **Foreign** option in the **Open File As** pull-down list. Then select the foreign format you want to use from the **Foreign Format** pop-up list. You set the information for foreign formats using another command, **Foreign Formats**, which is also found in the **File** menu. The **Define Format** button brings you to the **Foreign Formats** dialog without the need to exit the **Open As** dialog. See the description of the **Foreign Formats** command, below, for more information on how to use that command. When you click **Open** in the **Open As** dialog, the information contained in the definition of the foreign file format will be used to open the file.

### 7.2.4 Close Image

This command closes the active data window. If the data or CLUT has been changed since you last saved the image, then *IPLab* gives you the opportunity to save the window contents before closing it. This happens even if the **Close Image** command is executed from a script that you are running.

![Close Image Dialog Box](image)

In order to avoid this interruption, you may wish to use the **Dispose Window** command (**Window** menu). **Dispose Window** will not ask you if you want to save the image before it is closed.

### 7.2.5 Save

This command saves the contents of the active data window to its existing file. If the active window does not have an existing file (that is, if you didn’t open it and have not saved it before), then the **Save As** command will be executed, instead. This happens even if the **Save** command is executed from a script.

The **Save** command is used only to save images; it will not save scripts. If you execute this command while a script window is active, the script will record the **Save** command. To save a script, use the **Save As** button on the **Script Commands** palette.
7.2.6  Save As

This command saves the contents of the existing data window to a new file. A dialog box will prompt you for the new file’s destination, name, and format before writing the contents of the active data window to disk. If you enter the same name as an existing file in that directory, then you’ll be given the chance to cancel the action or to overwrite the existing file.

If you were trying to use the Save command to save an image for the first time, then this dialog box will appear to prompt you for a destination, file name, and format.

You may also specify to save only the ROI from the image and save a small preview image with the file.

![Save As Dialog Box]

7.2.6.1  Saving a Single File

Simply put, select the target directory in which to save the data, enter the name for the file, pick a file format, and click Save. The Single File radio button, which you use for the everyday saving of images, will become selected as soon as you click on a folder or begin typing a file name.

To save the file, navigate through your computer’s folders (directories) using the Desktop button, the Open button (which will replace the Save button when you click on a folder in the list), and the directory pull-down bar (which says Images in the example, above). When you click on the Desktop button, the file list will show the folders and images on your computer’s desktop. To see what’s inside any folder, double-click on it or select it and click the Open button. You can also move up in the folder hierarchy by selecting a higher folder from the directory pull-down bar.

When you find the folder in which you want to save the data, type the file name in the box under Single File. Select the file format using the Format box, described below, and then click Save.
7.2.6.2 Saving with a File List

Saving multiple files with the Use File List option lets you specify ahead of time the file names and destination folder for each of the saved files.

The file list is a list of the file names to give to the images to be saved; each file name has a destination folder associated with it. When you use the Use File List option, the file name and destination information for active window will come from the currently selected item in the file list. The next item is automatically selected in the file list, so that repeatedly using Save As: Use File List can save images with each of the file names in the list.

![File List Example](image)

Saving a File Using a File List

To use this option, you must first create the file list. Please see the description of the Edit File List command on page 169; it describes how to make a file list for use with the Save As command. Having made the list, make sure to select the entry in the list (probably the first entry) that you want to use with the first image saved. Also, select the right file list to be used with the Select File List command.

Then, when you are ready to save an image, choose the Save As command and click the Use File List radio button. The name of the file list will appear (in the example dialog fragment, above, the selected file list is "MPG List 1".) Choose the file format to use and click Save.

Save As cannot overwrite already-existing files with the Use File List option. If the image is to be saved in the same directory as another file of the same name, no data will be written and an error message will alert you to the problem. If this happens during the running of a script, the error message will suspend the script.

7.2.6.3 Saving an Indexed File

When you want to save a number of images with sequentially numbered file names, click the Indexed File radio button. Save As with the Indexed File option will save the active window with a base name and an index number. Each time you use the command, the same base name will be used but the index number will be incremented.

![Indexed File Example](image)

Saving an Indexed File

To use this option, you should first choose the Set Indexed Name: for Save command, which will set the base name, the starting index number, and the destination folder where the images will be saved. That command is described below, within this File menu section.

When you are ready to save, choose the Save As command and click the Indexed File radio button. The base name and current index number will appear on the dialog. (In the example dialog fragment shown above, the indexed file name is "Control Set 0000", where Control Set is the base name and the four zeros are the first index number.) Choose a file format and then click Save.
Save As cannot overwrite already-existing files with the Indexed File option. If the image is to be saved in another file of the same name, no data will be written and an error message will alert you to the problem. If this happens during the running of a script, the error message will suspend the script.

7.2.6.4 Saving the ROI Only and a Preview

If you check the Save ROI Only option, then when you click the Save button, only the data within the selection box will be written to disk. This keeps the file size down and saves you the trouble of cropping the image before saving it.

If you check the Save Preview option, then a smaller version of the image will be saved along with the full image. This small preview image can be displayed by the Open command to help you recognize the image when you are looking for it.

7.2.6.5 Choose a File Format

Before saving an image, you should pick which file format you want it saved as. This will affect what information is saved along with the image data, as well as how easily you will be able to view the saved image in other programs. The file format choices are TIFF, PICT, Text, FITS, EPR, Raw, and our own IPLab format.

Which file format should you choose? Please consult the information on page 128.

7.2.6.6 QuickTime Compressed PICT Files

If you have QuickTime installed, IPLab can QuickTime-compress PICT files. From the pull-down list of available formats, select the PICT (QT) option. The Set QT button will become active and the current compression setting will be shown in the Save As dialog below the Format box.

![Save As Dialog Box with PICT(QT) Format Option Selected](image-url)
By clicking on the **Set QT** button, you will get a second dialog which allows you to specify compression settings. You will also see a portion of the image to help you in determining suitable settings.

![Set QT Dialog Box](image)

**Set QT Dialog Box**

Only Apple Computer provides the QuickTime compression software; they include it as part of their operating system.

**Important:** You must note that saving image in PICT(QT) is a “lossy” method. That means that you sacrifice data integrity in order to achieve a smaller file size. You can no longer make perfectly quantitative measurements of the data. When you save images this way, the image that is left on the screen is not changed to reflect the true appearance of the image in the file. If you want to see the effects of compression on your image data, you must open the image file you just saved. If the results are not satisfactory, you should try saving the file again with alternative compression parameters.

### 7.2.6.7 Scripting the Save As Command

If you select either **Use File List** or **Indexed File** while making a script, the term `<scripted>` will appear where the name of the file list or indexed file would normally appear. This is because the name of the file list or indexed file is determined when the script is run.

In addition, when scripting, a **Use Window Name** checkbox appears next to the file name box. This option lets you default the name of the file to be whatever the active window name is at the time the script is run.

### 7.2.7 Revert to Saved

The contents of the active window are replaced with the data in the most recently saved version of the window. All changes to the original window are lost. If there is no saved file for the active window, an error message will be shown.
7.2.8 Delete File

Since image data requires a lot of disk space, you may find yourself in a situation where there is not enough room on the disk to save another file. The **Delete File** command lets you remove files from your hard drive from within the **IPLab** interface.

The **Delete File** command allows you to review the names of files on a disk and delete unneeded ones. A dialog will prompt you for the name of the file that you wish to delete. Then simply click the **Delete** button. The **Delete File** command will always prompt you to confirm that the selected file is to be deleted (though not when the command is scripted).

If you select the **From File List** radio button, the name and location of the file to be deleted will be obtained from the current location on the named file list. You must use this feature very carefully since other commands move the current file list pointer when they are executed! In order to use the **Indexed File** option, first use the **Set Indexed Name:** for **Delete** command (File menu). That will give the **Delete File** command the path and file name for the file to be deleted. No other commands can alter the index or base name used for deleting files.

Please see the description of the **Save As** command on page 161, since the dialog boxes for the two commands behave in the same ways.

7.2.8.1 Scripting the Delete File Command

There are several items to note when you select the **Delete File** command while scripting.

1. When **Delete File** is run from a script, you will **NOT** be prompted for a confirmation before the file is deleted! Such a request for confirmation would badly interrupt the script. Be very careful when using the **Delete File** command in a script, lest you delete your data.

2. If you select either **Delete File: from File List** or **Delete File: Indexed File** while making a script, the term `<scripted>` will appear where the name of the file list or indexed file would normally appear. This is because the name of the file list or indexed file will be determined when the script is run.

3. If your script has a **Delete File** command in it, you can bring up the dialog of that command to see what file you have selected. If you chose to delete a single file, the file name you have selected appears next to the radio button that is labeled **a Single File**.
4. **Important:** Suppose you choose the **Delete File** command, select the file "Washington", and click **Delete** to place the command in a script. You then go back and double-click on the **Delete File** command in the script to re-display its dialog. The highlighted filename in the dialog’s scrollable file list may not match the one you had selected. If you wish to accept the filename you have already selected ("Washington"), you must be sure to click on the **Cancel** button in the dialog. If you click on the **Delete** button at this point, the highlighted file in the dialog’s scrollable file list ("16Bit XRay") will be selected instead.

### 7.2.9 Export

This hierarchical menu allows you to export certain data views and images sequences in forms that may be read by other programs. The first two choices, **Export: View to File** and **Export: View to Clipboard**, can be used to save pictures of QGraphs and perspective views. The third choice, **Export: Sequence to Movie**, can be used to make a QuickTime movie.

#### 7.2.9.1 Export: View to File

When looking at data in QGraph and perspective views you may want to save a picture of the view itself, rather than the data used to create the view. You could then use the picture in publications and presentations. Choose **Export: View to File** to save a QGraph or a perspective view as it is displayed on the screen. Because the **Save As** command would simply save the data used to create the view, you must use the **Export** command. Please note that this command saves a picture, which will be treated like any other image if you open it in **IPLab**; it will not behave like the original graph or perspective view.

The **Export: View to File** command is only available when the active window is a QGraph or a perspective view, or when you are recording a script.

![Export: View to File Dialog Box](image)

Choose the folder where you want to save the view, type the name of the view’s new file (where the name "My Graph" is in the example dialog), and then click **Export**. The view can be saved as a TIFF, PICT, or PICT (QT). The printing quality of QGraphs is much better when they are saved in PICT format, so we recommend that you use PICT whenever possible. Perspective views saved as PICT produce smaller files than when saved as TIFF, but are otherwise similar.

If you need tips on navigating to the destination folder, please see the description of the **Save As** command (page 161), which has a similar dialog box.

When this command is scripted, a **Use Window Name** checkbox will appear next to the file name entry box. When **Use Window Name** is checked, the view’s new file will be given the name that the view currently has.
7.2.9.2 Export: View to Clipboard

When you choose the Export: View to Clipboard command, a picture of the currently active QGraph or perspective view will be placed in the clipboard. You can then switch to another application and paste that picture into another document. This command is only available when the active window is a QGraph or a perspective view.

7.2.9.3 Export: Sequence to Movie

This command will turn images into a QuickTime movie, which can be viewed on any computer (including PCs) that has a QuickTime movie viewer.

![Export: Sequence to Movie Dialog Box](image)

**Export: Sequence to Movie** Dialog Box

**Important:** The images must all be the same size and data type (e.g. all 800 x 600 pixels of type Unsigned 16).

The **From** options prompt you to identify the images you want to turn into a QuickTime movie.

**Active Window:** When this radio button is selected, the active, or front, window will be used to make the QuickTime movie. Use this option with multi-frame images sequences.

**Window:** Use this pull-down box to pick the name of an existing image window that will be turned into the QuickTime movie. You can also type a name, which is useful when scripting the command. This option, too, is for use with multi-frame image sequences.

**ROI Only:** This checkbox works with both the **Active Window** and **Window** options. When **ROI Only** is checked, only the selected region of the image window will be used in making the movie.

**Current File List:** When this option is used, **IPLab** uses the images listed in the selected file list to make the movie. If you have not already made a file list, then first choose the **Edit File List** command from the **File** menu. The name of the currently selected file list will be displayed on the line beside the **File List** heading.
Select File List: You can use this button to choose which file list you want to use. This button brings up the same dialog box as the Select File List command (File menu).

Indexed Files: With this IPLab radio button selected, IPLab uses the information in the Set Indexed Name dialog box to find the images with which to make the movie.

Set Indexed Name: Use this button to set the base name, index number, and location of the files to be made into a movie. This button raises the same dialog box as the File menu command, Set Indexed Name: for Open.

Save In: Use the Save In button to pick the destination folder where the QuickTime movie will be saved, and to give the movie a name. The name of the file and the path of enclosing folders will be displayed beside the Save In button.

Frames: Here, you can choose which frames of your image sequence will be used to make the movie. Click the All radio button to use all of the frames present in the image sequence, file list, or index sequence. Click the next radio button, and enter Start and Stop frame numbers, to use only some of the frames present.

Frame Duration: This controls the speed of the movie, or how fast the frames will go by in the movie.

Seconds/Frame: When this radio button is checked, each frame will be visible for the number of seconds that you enter in the text box. You can enter numbers smaller than 1 second.

Use Acquire Timing in: This radio button lets you vary the rate at which the frames appear. You must first create, and leave open, a timing window, or let the Create Timing Window feature of the Full Acquire command do it for you. Then select the name of the open timing window in this option’s pull-down box. You can also type in the window name, which is useful when scripting. When you click OK, the movie will be made so that the frames appear at the same rate as the rate at which they were acquired.

In order to make your own timing window (to completely control your movie), follow these guidelines. Make window of Floating Point type (with New Image), 1 pixel wide and with the same height as the number of frames you will be using. View it as text (with the View menu command, View As: Text). Enter the relative times, in seconds, by selecting each number and entering your own values with the Set ROI Value command (Edit menu). The times must be positive and must increase from top to bottom.

Compressor: You can pick from a number of different compression algorithms.

Quality: This pull-down box lets you pick the level of quality of the final, compressed movie.

Depth: This pull-down box tells the Export function what bit depth to make the final movie. Setting this to Image Depth will use the same bit depth as the precursor image(s). You can also set this to 8 (Byte), 16 (Short Int or Unsigned 16, or 24 bits (color).

Cancel / OK: When you click OK, the QuickTime movie will be constructed and saved in the location you set with the Save In button. You must use a QuickTime movie player to view your movie; one is available from Apple Computer, of course. If you click the Cancel button, then the dialog will disappear and no movie will be made.

Important: You must note that, depending on compressor, saving image sequences in QuickTime movies can be “lossy”. That is, compression typically loses or alters data in order to make the file smaller. This means that you sacrifice data integrity in order to achieve a smaller file size. IPLab automatically “flattens” the final movie to be played with QuickTime on Windows.
7.2.10   Edit File List

7.2.10.1   What is a File List?

A special feature of IPLab is its ability to process a collection of data files as a batch. IPLab offers two ways of accessing a collection of files, through indexed files (please see the Set Indexed Name commands, described starting on page 172) and through files lists.

The file list stores the names and locations of the files you wish to process. No matter what method you use to enter a name into the file list, information about the location of the file, including the folder and volume name, is also stored in the list. This information is always obtained from the folder and volume which are shown on the right side of the dialog at the time you enter the name (e.g. "IPLab Tutorial Images" in the example figure below, on page 170). You cannot change the path which is stored with a name once it is on the file list; however; you can delete the name from the list, change the directory in the right hand list then re-insert the name on the file list.

Important: IPLab will be able to use the file list to find the file so long as you do not change the file's name or directory location.

The commands which can use file lists are the Open, Open As, Delete File, Save As, and Export: Sequence to Movie commands, all from the File menu. When one of these commands is executed with the File List option selected in their dialogs, the command will obtain the name and location of the file from a file list rather than from any information entered in the dialog.

A file list is very powerful when used with one of the above commands within a script. In order to apply the same sequence of operations to a collection of files, all you need to do is:

1. Place an Open command which uses a file list at the beginning of the script, and
2. Place a Loop command at the end of the script which loops back to the beginning of the script.

A simple example of such a script is this slide show script:

Select File List   (Specify which file list to use)
Top                (Top of the loop)
Open               (Use the file list option)
Show Image        (Ensures proper image update)
Pause              (Use a long default Delay, such as 3600, i.e. 60 seconds)
Dispose Window    (Remove the image Window without prompting for OK)
Loop               (loop back to the line labeled Top)

Each time the file list is accessed by the Open command within the loop, the file list’s internal pointer is incremented to point to the next file in the list. Thus, each time the script goes through the loop, the next file in the list is opened and operated on.

Alternatively, when capturing a sequence of images with your camera or frame grabber, you may want to use a file list to save the images with special names. If your processing steps create temporary files on your hard disk, you can use a file list to automatically delete those files after they have been used.

Each file list can have a maximum of 300 entries. However, you may have any number of special files lists, each with a different name. So, for example, you may have different file lists for accessing all of the images you captured for your experiments on different days, or under different conditions.
7.2.10.2 Creating and Editing a File List

Use the Edit File List command to create and edit a file list. The Edit File List command displays a dialog with two scrollable lists.

The list on the left is the list of images in the file list. The pull-down box above the left hand list holds the name of the file list. The Add text entry box below the left hand list is for typing in file names to add to the file list.

The list on the right is the list of images in the current folder. The pull-down bar above the right hand list shows the current folder and can be used to navigate folders.

**Edit File List Dialog Box**

To learn to use this command, please use the options in the order listed below. You can skip many of these options, which you will not need. Basically, you will be adding file names from the list on the right to the list on the left, and clicking Done.

**File List’s Name:** The pull-down box in the upper-left corner lets you either select the name of an existing file list or type in a new name. Either way, you can then edit the contents of that list.

**File List Display:** The scrollable list on the left displays the image files that are in the file list. The selection bar (over "Washington", in this example dialog box) shows the position of the file list pointer, indicating the next file to be acted upon. This is the filename which is used when the file list is next accessed. Before executing a command which uses the file list, be sure to check that the desired filename is highlighted as the current item.

**Current Folder Bar:** The pull-down bar in the upper-right corner shows you the name of the current folder. The image contents of that folder are displayed in the scrollable list below. ("IPLab Tutorial Images" is the current folder in the dialog box, above.) You can use this folder bar to navigate upwards in the folder hierarchy.

**Existing Images on Disk:** The scrollable list on the right displays the images that are in the current folder. Select an image in this folder so you can use the Add button to add it to the file list. "Washington" is the currently selected image in the dialog box, above, and you can see that it was just added to the left hand list.

**File Format:** The This File field displays the file format of the currently selected image in the right hand list. The Show Type pull-down bar lets you control which file formats are shown. That helps you search for only one file format.

**Desktop:** Click this button to move to the top of the folder hierarchy, and to show the images stored on your desktop.
Eject: If you are scanning the images on removable media like Zip disks or floppy diskettes, use this button to eject the disk so you can insert another.

Add, Add All: These two buttons in the middle of the dialog box let you add image files from the right hand list to the left-hand list. When you click the Add button, the file selected in the right hand list will be added to the left-hand list. You can also just double-click on a file in the right hand list. When you click Add All, all of the files on the right will be added to the left. Files are added to the left-hand list above the currently selected item.

Add Edit Box: The Add button and edit box at the bottom-left of the dialog box let you type in a file name and add it to the file list. In this way you can even enter names of files which do not exist at the time you are creating the file list. This is useful, for example, when you expect to run a script which creates or saves a window which has to have a specific name.

Remove, Remove All: When you click Remove, the file listed in the left hand list will be erased from the file list. When you click Remove All, the entire list will be emptied, letting you start over.

Show Path: Click this small button, located below the left-hand list, to see the folder path of the file list’s currently selected file. You cannot change this path, but if necessary, you can remove the file and replace it with the correct one.

Done: When you are done adding files to the file list, click Done. This will save the file list and close the dialog box.

Cancel: Clicking this will close the dialog box without saving any of your changes.

This command is not scriptable. However, the next command, Select File List, is scriptable and enables scripts to switch file lists.

7.2.11 Select File List

You may have many different file lists. While you use the Edit File List command to create and edit file lists, you use Select File List to select the list to be used by the Open, Open As, Save As, Delete File, and Export: Sequence to Movie commands.

The File List pull-down box lets you select which file list to use.

Click the Reset to Top of List checkbox if you wish to start from the top (first) entry in the file list the next time you use it. This command is scriptable, so you may switch between file lists in the middle of a script.

The Store File Count... option stores the numbered position of the file list’s currently selected item in a variable. This is a useful way for other parts of your script to keep track of which frame is open. Simply check the box and enter the number of the variable in the text entry box.
7.2.12 Set Indexed Name

Indexed files have a base name and a count, or index value. Examples of indexed names are "New Study 0000", "New Study 0001", etc. Indexed files are used by the Open, Open As, Save As, Delete File, and Export: Sequence to Movie commands. To make it easier to manage a large number of files, these commands can work on one file after another, in the numerical order of their file names. This lets you process a large number of files in sequence.

Separate base names and index numbers are maintained for opening, saving, and deleting. So, for example, when you open an indexed file, the index values for Save As and Delete are not affected. The three commands for setting the three separate base names and index numbers are Set Indexed Name: for Open, Set Indexed Name: for Save, and Set Indexed Name: for Delete. The dialog boxes look the same, however:

![Set Indexed Name: for Open Dialog Box](image)

**Base Name:**

The base name will be the first part of the file name and will be the same for all of the indexed images.

When the Date radio button is selected, today’s date will be used as the base name. You can set the date format (e.g. 2001/12/14 or 12/14/01) using the Preferences command.

When the second radio button is selected, the text you enter into the text entry box will become the base name. The index number will immediately follow the base name, so if you want a space between the two, add a space to the end of the base name.

**Index:**

This dialog box allows you to set the starting index to a specific number (when the #/Var icon shows #), or to the value of a specific IPLab variable (when the #/Var icon shows Var). The index value (not the variable number) which will be used is shown in parentheses as “currently…” If you use the Var option in the #/Var button, the value of the specified variable will be incremented each time the applicable command is executed.

**Reset to 0:**

Pressing the Reset to 0 button will reset the index to zero. This button is grayed out when the index is stored in a variable.

On the right side of each Set Indexed Name dialog is a scrollable list which shows a Macintosh file folder. The selected folder is the one in which the Open command will find its files; in which the Save As command will put its files; and in which the Delete command will find the files to be deleted. That file folder information is stored with the index base name so that all subsequent uses of the indexed name will use the correct folder.

The Set Indexed Name commands are scriptable, and you can change the base name and index values at any time.
7.2.12.1 Example Usage

Here is an example of when you might use indexed files: You are writing a data acquisition script, and you want the file names to be similar for this group of images. You will also likely want to close each image after saving, in order to save memory space. Using the **Set Indexed Name: for Save** command, you set up a base name, “New Study”, make certain that the index number is zero, and select the folder in which to save the data. In your script, each acquisition is followed by a **Save As** command with the **Use File List** option selected.

7.2.13 Foreign Formats

**IPLab** can read image files with data saved in a wide variety of formats. You use this command to tell **IPLab** about these formats. We call these “foreign files.” In the dialog for this command, you specify a name of your own choice for the foreign file format, plus the details of that file format. You refer to these names in the **Open As** command when you select **Foreign** in the **Open File As** pop-up list of file types. You can bring up this same dialog box by clicking the **Define Foreign** button in the **Open As** dialog.

![Foreign Formats Dialog Box](image)

In order to read the data contained in foreign files, **IPLab** makes some assumptions about the file format. **IPLab** will read foreign data of type Byte, Short Integer, Unsigned16, Long Integer, Floating Point (IEEE format floating point only), and Color 24. It is assumed that the file may contain a header which is followed by the image pixel data. The data is assumed to be stored as a sequence of horizontal lines of the image, where each line has one or more terminating bytes at the end (that is, end of record marks). This is a fairly common format for exchanging image data outside the Macintosh community, so you generally should have no problem obtaining the information you need in order to read such files.

The Macintosh stores Short Integer and Long Integer data with the lowest order byte at the highest memory location within the data word, while some other computers store Short Integer and Long Integer data in the reverse order. In order to read data which has been stored by those computers, **IPLab** also lets you specify whether to reverse the order of the bytes in two or four byte/pixel data once it is read.

If you are importing color image data, you must know something about the order in which the color components are stored. If your foreign color image data is stored so that the red, green and blue components for each pixel are stored together (i.e. the pixel data is stored like RGB, RGB, RGB, ... ) then select the **Interleaved RGB** check box. If your foreign color data is stored so that the red, green and blue components for each pixel are stored sequentially (i.e. all of the red pixel values, followed by the green pixel values, followed by the blue pixel values) then leave that check box unchecked. When either of the color types is selected, **IPLab** sets the number of bytes per pixel and resets the **Invert Byte Order** checkbox. **Invert Byte Order** is not necessary for color images.

Do not worry if your image data is stored with the red, green and blue components in an order different than R-G-B. Go ahead and read the image data. Next, use the **Split Color Channel** and **Merge Color Channel** commands to split the image up into its color components and put it back together properly.
Most binary data files contain the size information in their headers. If you know where in the header of your foreign file the width and height information is located, you may instruct IPLab to get that information directly out of the header. Check the box marked **Size From Header** and enter the **Byte** locations (the number of the byte in question, numbered from the start of the file, with the first byte in the file equal to byte 0) for the **Width** and **Height** parameters. The **Width** and **Height** parameters will most often take up two bytes in the header, but you may select 1, 2 or 4 for the **# Bytes** parameter. If the **Invert Byte Order** box is checked, IPLab also inverts the byte order when reading the **Width** and **Height** parameters.

If you do not know the header length or the number of extra bytes/line, start with zero values for these parameters. If the total number of bytes you have specified is not the same as the file size, an alert will tell you so. The **Open As** command will display an alert to tell you about the discrepancy and to give you the option of either canceling the operation or letting IPLab proceed to display the data as specified. Discrepancies may be due to an error in any one of the parameters: header length, pixels per line, number of lines, extra bytes/line, or data type. The most common errors will be in the header length and the number of extra bytes/line.

If you see most of the foreign image displayed correctly, then the error is probably due to an incorrect header length parameter or to the wrong number of lines. An incorrectly specified header length often makes the image appear to be circularly shifted in the horizontal direction. Otherwise, the specified number of lines is probably wrong. If you display an incorrectly-read image, dispose of the image window, choose the **Foreign Format** command again and adjust the header length or number of lines parameter as appropriate. This time when you open the file, the alert should not appear.

If the foreign image appears distorted as if it were slanted, then either the number of extra bytes/line or the number of pixels per line is wrong. If you are sure of the line length, adjust the number of extra bytes/line.

An image which appears to contain diagonal lines slanted from the upper-left to the lower-right contains more extra bytes/line than you specified. In this case, add one to this parameter for each of the diagonal lines you see in the distorted image. (The example figure below illustrates a case in which 2 extra terminating bytes per line were needed.) An image which appears to contain diagonal lines slanted from the upper-right to the lower-left contains fewer extra bytes/line than you specified. In this case, subtract one from this parameter for each of the diagonal lines you see in the distorted image.
As you experiment with these parameters, you will find that you can read a large variety of images from many different sources.

**Note:** If you want to read only the top portion of an image, and really know what you are doing, you may not want the alert to tell you that there is a discrepancy in the file size. In such a case, just put the **Open As** command into a script and run the script. The alert does not appear when running a script.

Alternatively, you may want to read only the last portion of an image. When you set the **Skip Lines** parameter to value \( n \), **IPLab** first skips past the header and the first \( n \) rows of the image data before beginning to read the image. The image data is always read into a new image window that has a height equal to the **Number of Lines**. To avoid the discrepancy alert, the sum of the two parameters **Skip Lines** and **Number of Lines** must equal the total number of lines in the image. Therefore, if your image has 2048 lines, you may read the second half of the image by setting **Skip Lines** = 1024 and **Number of Lines** = 1024.
7.2.14 Edit Image Info…

The **Edit Image Info** command lets you type descriptive information for the active (front) image window. Where the **Define Image Info** command defines the information for all new images, the **Edit Image Info** command defines the information for the existing, currently active image.

![Edit Image Info Dialog Box](image.png)

**Edit Image Info** Dialog Box

The **Short Description** field is good for storing information about a group of images, such as the date on which a group of images were acquired, or the name of the researcher who acquired them. The **Short Description** field can contain up to 63 characters.

The **Notes** field is good for storing information specific to this single image. The **Notes** field can contain up to 255 characters.

You can view the active image's information in the **Edit Image Info** dialog box by typing Command-i (⌘-i).

### 7.2.14.1 Scripted Behavior

When you script the **Edit Image Info** command, checkboxes appear next to the **Short Description** and **Notes** fields. Check the boxes for the fields you want the scripted command to change. Uncheck a box to leave that field unchanged.
7.2.15 Define Image Info…

The Define Image Info command defines descriptive information that will be stored with all new images. Descriptive information for your data can include, for example, the date and the identities of the sample, probes, and researcher. All new images created by acquisition commands and by the New Image command will have this text stored in their image info.

![Define Image Info Dialog Box]

The Short Description field can contain up to 63 characters.

In the example dialog shown above, the date has been typed into the Short Description field. All newly acquired images or images newly created by the New Image command will contain this date in their image info. You could then view the acquisition date of the images in the Edit Image Info dialog box by typing Command-i (⌘-i). On each new day, you would update the Short Description information.

7.2.16 Edit FITS Header

FITS, which stands for Flexible Image Transport System, is commonly used to exchange image data among different computers. The FITS header contains standardized information about the image. The basic FITS header consists of 5 lines (plus the END line) that are always present. You cannot edit or delete those lines.

The Edit FITS Header command can, however, add (insert) and edit new lines within the header. For example, IPLab adds acquisition information to the FITS header when you save your data in the FITS format after acquisition. You can edit and delete those lines. You can also add, edit, and delete your own lines.

![Edit FITS Header Dialog Box]

To insert a new command, you must select either the END line or a line other than one of the standard first five, and then click Insert. To edit or delete a line other than one of the standard five, select the line and click Edit or Delete.
The **Edit FITS Card** dialog will open:

![Edit FITS Card Dialog Box](image)

**Edit FITS Card** Dialog Box

It is your responsibility to ensure that the information you enter is in correct FITS format.

Please note that the **Edit FITS Header** command is not scriptable.

### 7.2.17 Page Setup

This command allows you to set up the parameters for printing the active image window. The dialog which appears depends on which version of the printer driver you are using. Below is an example dialog for a LaserWriter.

![Page Setup Dialog Box](image)

**Example of Page Setup** Dialog Box, with **IPLab** Options

**IPLab** adds several options to the **Page Setup** dialog. In earlier versions of the driver, the options were added to the bottom of the main dialog. In later versions, as illustrated here, the options appear in a separate screen, which is reached via the pop-up menu at the top. **IPLab** adds check boxes that let you print a title line at the bottom of the page, print only the data within the image’s selection box, and adjust the size of the printed image so that it fills the page on which you are printing. The title line includes the date and time of the print command, the window title, and the page number.
There is also a **Set Size** button that brings up another dialog for adjusting the print resolution.

![Set Size Dialog Box](image)

**Set Size** Dialog Box

The top part of this dialog describes the resolution parameters that were saved with the image file. You can set any value you want for the print resolution in this dialog, within the available resolutions for your printer. You can also change the dimensions of printed image and let **IPLab** compute the resolution. Changing the print resolution or image size does not affect the image data in any way.

### 7.2.18 Print

The **Print** command prints script windows and image windows. The result of the print command for image windows depends on how the data is viewed:

<table>
<thead>
<tr>
<th>Viewed As:</th>
<th>What is Printed:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image (Including 1D)</td>
<td>An image (even if 1D)</td>
</tr>
<tr>
<td>QGraph</td>
<td>QGraph plot</td>
</tr>
<tr>
<td>Text</td>
<td>Text page</td>
</tr>
</tbody>
</table>

When you view 1D data as an image, *i.e.* as a simple plot, then **IPLab** does not print that view as a plot. If you wish to print a plot, you must first view the data as a QGraph, and then print.

Images that are larger than a single page are printed across multiple pages. The pages are printed from top to bottom and from left to right. So the top-left of the image is on the first page printed.

As with **Page Setup**, the dialog that appears is dependent upon your current printer driver: Earlier versions contain the **IPLab** additions at the bottom of the main dialog. Later versions, as shown here, display the options in a separate screen.

**IPLab** uses the printer driver that you last selected from the Chooser. It may also be necessary to also click on the checkbox entitled **Swap Black and White**. The standard Apple drivers for PostScript printers currently force black and white (only) to be backward from the representation used by **IPLab**. Using this option corrects the problem. Note that Apple’s QuickDraw printers do not seem to have this problem. You may need to experiment by printing a test image containing an area of pure black (0), pure white (255) and a gray ramp (use the **Math** menu command **Set Pattern** in a blank image). If the black and white areas are swapped, and the grays are okay, then you need to use the **Swap Black & White** option.

When a script window is open, the number of copies to be printed appears to be set equal to 1 regardless of the entry made in the **Print** dialog. Actually, the number of copies is remembered, but then reset to 1 if you re-open the dialog from the script. Because of this, the best way to print multiple copies of the same image is to follow the **Print** command by a **Loop** command with **# Iterations** set equal to the number of copies desired.
7.2.19 Quit

Before IPLab quits, you are asked if you want to save any windows which have not been saved since they were last changed. For each window which has not been saved, you may choose to save it, not save it, or cancel the Quit command.

The Quit command is scriptable. This leads to the common mistake of selecting Quit while a script window is active, and recording the Quit command instead of actually quitting. If you wish to quit while a script window is active, hold down the Option key while selecting the menu item, or while typing Command Q. This executes the Quit command instead of placing it in the active script.

If you hold down the Command, Option, and Shift keys all at once, and then choose Quit from the menu, or type a “Q”, IPLab quits immediately, without saving any files. Be sure that you have already saved any important windows before using this feature.
7.3   Edit Menu

7.3.1   Undo

Since image data requires such a large amount of memory, *IPLab* does not currently support an always-present *Undo* function. You may achieve the benefits of a manual *Undo* function by use of the *Duplicate Window* command in the *Window* menu and the *Revert to Saved* command in the *File* menu.

7.3.2   Cut

The *Cut* operation affects data windows. It is used to simultaneously copy information to a paste buffer and to clear the selected information.

When you use this command interactively, outside of a script, the information that gets cut depends on which tool you have selected on the *Tools* palette. If you have one of the *ROI* tools or one of the *Image* tools selected, then the image data within the ROI will be cut. First, the data within the ROI will be copied, as if the *Copy* command had been used (see the *Copy* command, below). Then the data values of all of the pixels within the ROI will be set to the value 0, leaving a dark hole.

If you have one of the segment tools selected, then all segments that are within the ROI will be cut, while the image data will be unaffected. In this regard, *Cut* works exactly like the *Cut Seg* tool on the *Tools* palette. If you have one of the drawing tools selected, then the highlighted (selected) drawing objects will be cut, regardless of the ROI. In this regard, *Cut* works exactly like the *Cut Draw* tool on the *Tools* palette.

If you are viewing the data as a QGraph, then when you select *Cut*, the entire graph will be placed in the QGraph buffer, but the graph will not be cleared. In this respect, *Cut* appears to work as the *Copy* command. The ‘cut’ graph may then be pasted into another application, but not into an *IPLab* window.

If you are viewing the data as a perspective, then the entire graph is placed in the perspective buffer and the data within the ROI is cleared. The graph may then be pasted into another application, but not into an *IPLab* window.

7.3.2.1   Scripting the Cut Command

When placing the *Cut* command into a script, a dialog appears which lets you specify what to cut.
You have three choices:

1. The image data within the ROI. This is what you normally cut when you use the Cut command directly on the image.

2. Segment information within the ROI. This is the same as using the Cut Seg button on the Tools palette.

3. Selected drawing items. This is the same as using the Cut Draw button on the Tools palette.

The dialog box also allows you to cut information from any window, and not just the active window.

### 7.3.3 Copy

The Copy operation applies to data windows. It is used to copy information to a paste buffer so you can place it in another window. When you use this command interactively (outside of a script), the information that gets copied depends on which tool you have selected on the Tool palette. If you have one of the ROI tools or one of the Image tools selected, then the image data within the ROI is copied.

If you have one of the segment tools selected, then the portion of the segment that is within the ROI is copied. In this regard, Copy works exactly like the Copy Seg button on the Tool palette. If you have one of the drawing tools selected, then the highlighted (selected) drawing objects are copied. In this regard, Copy works exactly like the Copy Draw icon on the Tool palette.

When you copy data, the copied “stuff” also goes to the Clipboard. The Clipboard is used to paste what you see on the screen into another application (such as another image processor, a word processor or a spreadsheet).

No matter what you copy (image data, segment data, or drawing objects), the information is copied to a paste buffer so you can transfer it to other windows within IPLab. The contents of the Clipboard and the data paste buffer depend on the way you are viewing the data at the time of the copy. When copying an image viewed as an image, then both the data and the CLUT of the image are copied into the paste buffer so that you may paste the sub-image onto another image of the same data type. IPLab treats Color 24 and Color 48 as distinct data types, so you cannot paste Color 24 data onto a Color 48 image or vice versa. If you wish to combine portions of several images which have different color types, you should first use the Change Color Type command (Math menu) to make each of the images type Color 24 or Color 48. After doing so, you may freely copy and paste between the various images.

<table>
<thead>
<tr>
<th>Viewing As</th>
<th>Clipboard contains</th>
<th>Data Paste Buffer Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image (either as 2D image or as 1D basic plot-not QGraph)</td>
<td>Image within ROI only if the ROI or Image tools are selected</td>
<td>Data values within ROI, or segment data, or draw objects, depending on the tool selected</td>
</tr>
<tr>
<td>Text</td>
<td>Text within ROI</td>
<td>Data values within ROI</td>
</tr>
<tr>
<td>QGraph</td>
<td>Entire QGraph plot</td>
<td>Unaffected</td>
</tr>
</tbody>
</table>
7.3.3.1 Scripting the Copy Command

When placing the Copy command into a script, a dialog appears to let you specify what to copy.

![Scripted Copy Dialog Box](image)

You have three choices of what to let the script copy:

1) The image data within the ROI. This is what you would normally copy if you use the Copy command directly on the image.

2) Segment information within the ROI.

3) Selected drawing items.

The dialog box also allows you to copy information from any window, and not just the active window.

7.3.4 Paste

The Paste operation affects data windows. Pasting places the contents one of the paste buffers in the active image. When you use this command interactively (outside of a script), the buffer that gets pasted depends on which tool you have selected on the Tool palette. If you have one of the ROI tools or one of the Image tools selected, then the data paste buffer that was last filled with image data within an ROI will be pasted.

If you have one of the segment tools selected, then the segment paste buffer that was last filled with segments will be pasted. In this regard, Paste works exactly like the Paste Seg tool on the Tool palette. If you have one of the drawing tools selected, then the draw object paste buffer that was last filled with drawing objects will be pasted. In this regard, Paste works exactly like the Paste Draw tool on the Tool palette.

The Clipboard also holds the information ready to be pasted into another application (such as another image processor, a word processor or a spreadsheet).

The action of the Paste command on data also depends upon whether the active window contains 1D or 2D data.
7.3.4.1 Pasting in 2D Data Windows

If the active window contains 2D data, then the contents of the paste buffer are overlaid on the active window provided that:

a. the data types of the paste buffer and the data window match (for example, one can paste Long Integer data only into a data window of Long Integer type); and that

b. the data in the window is not hidden (see Show/Hide Display in the View menu).

If the Paste command is executed interactively and you are viewing the data as an image, then you may drag the newly pasted data around the image window by pointing to it, holding the mouse button down, and dragging. Once you click the mouse anywhere outside the pasted data, the new data replaces the data in the pixels which it overlaps.

If the Paste command is executed by running a script or while you are viewing the image as text, then the location at which the information is pasted is specified by the ROI of the active window. In this case, the pasted data is placed so that the upper left corner pixel of the pasted data matches the upper left corner of the ROI, and the new data automatically replaces the data in the pixels which it overlays. This allows you to paste image data into one or more images at precise locations while executing a script and without interrupting the script operation. Note that the ROI in the active image window is only used to define the location of the pasting. The pasted data may extend beyond the ROI. In fact, you need only select a single pixel in the active window in order to paste any size data paste buffer when the Paste command is executed by running a script.

After pasting into an image, the location of the ROI in the active window is changed to be the same as the paste region.

As you are pasting, IPLab converts the CLUT of the destination image into the CLUT in the paste buffer, then copies the data out of the paste buffer into your destination image. This way, the portion of the image which is pasted will look exactly as it did when you copied it, and the data values will be exactly what they were when you copied them. If you are working on Byte type (8-bit) images, and the CLUT of the destination image differs from the CLUT of the source image, then this CLUT change will make the rest of the destination image look strange. This is a universal problem with copying and pasting between Indexed color images. When the CLUTs do not match, it is not possible to make the Byte type images look right. (In fact, some applications simply will not let you do the paste in such cases.) Fortunately, the Change Color Type command (Math menu) lets you convert all your images to Color 24 or Color 48, where such CLUT incompatibilities do not matter.

7.3.4.2 Pasting in 1D Data Windows

If the active window is a 1D data window, pasting automatically overwrites data in the active data window so long as the active window is not hidden and the data types match up. You cannot move the data around the 1D data window before fixing the pasted area as you can for image windows. The pasting location for 1D data is always defined by the top-left corner of the ROI, whether the pasting is done interactively or from a script, and whether you are viewing as data or text. The affected data elements (pixels) start with the first element of the ROI and continue for the length of the paste buffer or until the end of the data is reached. The pasted data may extend beyond the ROI. In fact, you need only specify an ROI of size 1x1 in the active window in order to paste any length of 1D data, whether the Paste command is executed interactively or by running a script.

After pasting into a 1D data window, the location of the ROI in the active window is changed to be the same as the paste region.
7.3.4.3 Scripting the Paste Command

When placing the Paste command into a script, a dialog appears which lets you specify what buffer to paste from.

You have three choices:

1. The image data buffer. This is the image data that you normally paste when you use the Paste command directly on the image.
2. Segment data. This is the same as using the Paste Seg button on the Tools palette.
3. Drawing items. This is the same as using the Paste Draw button on the Tools palette.

The dialog also allows you to paste information into any window, and not just the active window.

7.3.5 Clear

The Clear operation applies to data windows. It is used to remove information, leaving zero-valued (dark) pixels behind. When you use this command interactively (outside of a script), which information gets cleared depends on which tool you have selected on the Tools window. If you have one of the ROI tools or one of the Image tools selected, then the image data within the ROI will be cleared, or set to zero.

If you have one of the segment tools selected, then the segments that are within the ROI will be cleared, i.e. it will be set to the transparent color. If you have one of the drawing tools selected, then the highlighted (selected) drawing objects will be cleared, i.e. they will be removed.

The contents of the paste buffers are not affected by the Clear command.

On Color 24 and Color 48 images, the Clear command sets the values of all three-color components to zero.
7.3.5.1 Scripting the Clear Command

When placing the Clear command into a script, a dialog appears which lets you specify what to clear.

![Scripted Clear Dialog Box]

You have four choices of what the script should clear:

1) The image data within the ROI. This is the image data that you normally clear when you use the Clear command directly on the image.

2) Segment area. This clears the pixels in the segment overlay within the ROI.

3) Selected drawing items.

4) Registration marks.

The dialog also allows you to clear information within any window, and not just the active window.

7.3.6 Crop Image

The Crop Image command reduces an image window to a smaller window that contains only the selected portion.

Select part of the image and then choose Crop Image. A new, smaller window will be made that contains only the selection. The window name will remain the same. The old window will be disposed of without being saved, and without warning you first. The new image will have the same number of frames as the original and will contain all of the data, segments, and drawings that were within the ROI.

The Crop Image command is scriptable.

To get more versatility, use the Duplicate Window command (Window menu).
7.3.7  Set Value

Use this command to fill part of the image with a single intensity value: black, white, or any shade of gray.

First, either select part of the image inside an ROI or segment the data. Then choose the Set Value command.

If you want to change the selected pixels, choose the Inside ROI radio button.

If you want to change the pixels under a particular color of segment, then click the Segmented As radio button. Choose the color of the segments by clicking on the color bar.

Finally, type the fill value in the To Value field. You may enter integer or floating point values for the fill value; however, if you enter values outside the dynamic range of the data type, the exact value used may be unpredictable.

When you use this command on color images (data type Color 24 or Color 48), the pixel values for the red, green, and blue channels will each be set to this fill value, producing a shade of gray. Use the Set Color command to replace the ROI with a color.

If you set the #/Var icon to Var, you will set the pixels within the ROI to the value of the specified IPLab variable.

7.3.8  Set ROI Color

Use this command to replace the selected part of an image with a specific color. This command sets the red, green, and blue components for each pixel in the ROI of the active window. Unlike the Set Value command, the Set Color command can produce a color other than gray. This command operates only on Color 24 and Color 48 image windows.

First, either select part of the image inside an ROI or segment the data. Then choose the Set Value command.

If you want to change the selected pixels, choose the Inside ROI radio button.
If you want to change the pixels under a particular color segment, click the **Segmented As** radio button. Choose the color of the segments by clicking on the **Segmented As** color bar.

Finally, click the **To Color** box to pick a color. The Apple Color Picker, which is part of the system software, opens:

The Apple Color Picker

You may find that the RGB Picker is the most straight-forward, since **IPLab** stores color images in RGB format. If you do not have the Color Picker extension in your System: Extensions folder, then **IPLab** will produce an HSV color picker similar to the one shown above.

### 7.3.9 Select All

The **Select All** command selects (makes active) all information in a data window. Actually, when you use this command interactively, the information that gets selected will depend on which tool you have selected on the **Tool** palette.

If you have one of the ROI or segment tools selected, then the entire window will become bounded by the ROI. The pixel data become selected.

If you have the registration mark tool selected, all of the registration marks will be selected (highlighted).

If you have one of the drawing tools selected, then all of the drawing objects will be selected (highlighted).
7.3.9.1 Scripting the Select All Command

When placing the Select All command in a script, a dialog will appear to let you specify what to select.

![Scripted Select All Dialog Box](image)

The first two radio buttons let you choose which window to use. You can select information from the active, or front, window. You can also use the second radio button to pick the window by name. If the window is not open yet (but will be when the script is run), then you can type the name into the Window text box.

The scripted command can select one of four things:

- **Image Data (ROI):** This sets the ROI to be the biggest rectangle possible around the entire image.
- **Segment Area:** This also sets the ROI to be the biggest rectangle possible around the entire image.
- **Drawing Items:** This selects all of the drawing objects as if you used the Select All Drawings command.
- **Registration Marks:** All registration marks become selected, letting you move or delete all of them.
7.3.10 Define ROI

This command allows you to create the ROI in one of several ways. It is an especially useful command to insert in a script when you want the script to create a ROI by itself.

The top two radio buttons let you choose which window to use. You can use the active window, or you can pick the window by name. If the window is not open yet (but will be when the script is run), then you can type the name into the Window text box.

The first six ROI options let you define the boundaries or coordinates (in image pixels) of your ROI in the text entry boxes on the right side of the dialog box. Remember that the image’s top-left corner has the coordinates (0,0), that x increases from left to right, and that y increases from top to bottom. Simply select the ROI option, enter the needed coordinates, and click OK. The ROI will appear on the image.

When you define a custom rectangle or oval, you can enter the bounds of the rectangle as the left, right top, and bottom parameters, or you can enter the location of the center of the rectangle or oval, and also specify its width and height. You can define a line by giving its two endpoint locations, or you can specify one endpoint and the distance in x and y to the other endpoint. Other options are to place the ROI around the selected drawing object or around the entire image.

Finally, the last two options define an oval or rectangular ROI with the same left, right, top, and bottom boundaries as the existing ROI.

You use linear ROIs, for example, when you wish to know the data values along a slice through your image (see the Extract: Linear ROI to 1D command in the Analyze menu). You can define a linear ROI using the mouse with the line tool, but accurate definition of the line may require using this command.

If you set the #/Var icon to Var for any of the parameters, that parameter value will come from the specified variable.

If you enter boundary values which result in an ROI larger than the image (or if the variables contain such values), then the ROI will be clipped to the dimensions of the data. If, for example, you want to process all of an image except the top row, but you don’t know the exact dimensions of the image, set the boundary parameters to:

Left = 0        Right = 32767        Top = 1        Bottom = 32767
7.3.11 Modify ROI

Use this command to change the size of the ROI or to move the ROI.

**Modify ROI Dialog Box**

Entering positive values in the Inset $\Delta x$ and $\Delta y$ boxes will make the ROI smaller; negative values will make the ROI bigger. If an Inset action would cause the ROI to pass the boundaries of the image, the entire Modify ROI command will fail and the ROI will not change or move.

Entering positive values in the Translate $\Delta x$ and $\Delta y$ boxes will move the ROI to the right and down; negative values will move it to the left and up. The ROI cannot be translated beyond the top or bottom edge. If a Translate action would cause the ROI to pass the top or bottom boundaries, any Inset actions will be allowed but the Translate action will fail. The same thing would happen at the left and right edges, unless the Wrap at Left/Right Edges option is checked.

The Wrap at Left/Right Edges option can be used to “scan” the image with a fixed ROI. When either the right or left edge is reached, the ROI is “wrapped” to the opposite edge and shifted down by an amount equal to the height of the region. Also, if there is not enough space to move the ROI to the edge of the image, then it will be wrapped to the opposite edge and down.

You can see that this lets you use Modify ROI: Translate to move the ROI over the entire image. When analyzing your data, you would use the Define ROI command to make a ROI that is a fraction of the image size. Since the image size is 256x256 pixels, the ROI has been made to be 64x64 pixels. That way, no part of the image will be skipped. After measurements were performed on the ROI, the Modify ROI command would be used to translate the ROI to the right, where measurements would be done again. After the right edge was measured, the ROI would be wrapped around and down so that a new portion of the image could be measured.
7.3.12 Delete Frames

Use this command to delete one or more frames from an image sequence. This command is only available when the front window contains multiple frames, or when you are scripting.

![Delete Frames Dialog Box](image)

The dialog lets you name the window containing the image sequence and specify the range of frames to be deleted. If you select the Var option in place of the # option for the frame number parameters, the range of frames will be obtained from the IPLab variables.

7.3.13 Transfer Frames

This command copies frames from one image sequence and pastes them in another image sequence, overwriting the previous data. This command is only available when the front window is an image sequence.

![Transfer Frames Dialog Box](image)

In the From fields, choose the source window, which may be the active window or any other open window. Also choose the frames that you want transferred. In the example dialog box above, three frames are being transferred: frames #0-2.

In the To fields, choose the destination window. You can either create a new image sequence by clicking the New Window radio button, or you can overwrite frames within an open sequence picked from the To: Window field. Also choose the frame number for the transfer, unless you're creating a new window. In the example dialog box above, frames #0-2 are being transferred to frame #23, which means they will replace frames #23-25.

If you select the Var option in place of the # option for the frame number parameters, the range of frames and the starting destination frame can be obtained from variables.
7.3.14 Reverse Sequence

The Reverse Sequence command simply reverses the order of frames in a sequence. It can be undone by applying it a second time. The command always operates on the front-most window. There is no dialog for this command. This command is only available when the front window is an image sequence.

7.3.15 Change Drawings

With this command, you can change the color of selected drawing objects.

Click the color bar to pick the new color for the drawing(s).

7.3.16 Clear All Drawings

This command removes all drawing objects from the overlay of the active window.

7.3.17 Select Next Drawing

Use this command to select drawing objects in the active window. This command selects drawings in the order in which they were drawn. If no drawings are initially selected, Select Next Drawing will select the first drawing.

If a drawing tool has not been activated on the Tools palette, then

You will only see the selection line around the drawing object when a drawing tool is active on the Tools palette. With or without a visible selection line, however, the object will still be selected.
7.3.18 Transfer Attributes

This command transfers selected image display settings and overlays (attributes) from one window to one or more other windows.

**Transfer Attributes** Dialog Box

You can transfer one or more of the attributes listed next to the checkboxes. The **From** and **To** windows must be the same size in order to transfer the ROI definitions or segment layers. If you select **To: All**, all open windows will be affected. If a window cannot accept the attribute (for example, because it is the wrong size, or does not have a segment layer), it will be ignored.

One common use of this command is to normalize two images the same way, so that pixels with the same value will appear with the same intensity and can be visually compared. To do this, select the two images in the **To** and **From** sections. Check the **Normalization Values** checkbox. When you click **OK**, the normalization values being used for the **To** image will be applied to the **From** image, and pixels with the same value will appear the same.

To transfer segments between images, select the **Segment Layer** option in the dialog. You can use this feature, for example, to analyze the same regions in multiple images in a study: Define specific regions to analyze in one image using segments, and then transfer the segment layer to all of the other images in your study. You can then measure exactly the same regions in each image.
7.3.19 Set Row/Column Label

When you are viewing data as text, you may find it very helpful to add text headings to the rows and columns of numbers. You may have noticed that IPLab creates some of these headings for analysis results, for example. This command lets you add those labels, too.

Set Row/Column Label Dialog Box

To use this command, click either the Row or the Column radio button, and then enter its number. Enter the text of the label in the long text box. You can use the Apply button to apply the changes you make without closing the dialog. This lets you interactively change the labels on multiple rows or columns.

This command applies only to the active window. It may be used independently of how the data is viewed, but of course you can only see the labels when you view the data as text.

7.3.20 Stamp Overlays On Image

This command transfers the segments and drawings from the overlay layer to the image itself. It also removes the overlays from the image. This replaces some of the image data with the overlay information.

You may have added annotations to your images with the drawing tools, for example. If you want to see those annotations using another graphics program, you will need to stamp the overlays into the data. Use the Stamp Overlays on Image command, save the image as a TIFF or PICT, and then open it up in the other program.

Be careful when using this command because it alters the data values of the image.
7.3.21 Assign F Keys

This command allows you to assign menu commands and scripts to the function keys found on most keyboards. Once the keys are assigned, pressing the key will execute the command or script. This is an excellent way to automate IPLab for use when your attention is fixed on your microscope or experiment.

Assign F Keys Dialog Box

To assign a command to a particular key, click on the key name (e.g. the F6 button that’s highlighted above), and then select a menu command, click a tool on the Tools palette, or run a script. Any existing command will be replaced by the new one. To remove a command assignment from a key, click on the key name and press the Delete key on the keyboard. When you click OK, the command assignments will be saved in the IPLab User Data file.

Menu commands are any command which may be selected from the menu bar, including the Control and Ext. menus. Pressing a function key that has been assigned to a menu command executes that command just as though it had been selected from the menu bar. You may also select any of the ROI, segmentation, or drawing tools from the Tools palette. When the function key for a tool is pressed, the result will be the same as though you had clicked on the Tools palette.

To assign a script to a function key, select Run Script from the Script menu and then select a script to be run. The name of the selected script is drawn in bold letters next to the function key name. When you press the key, the script will be run as if you had selected Run Script from the File menu.

You can also assign commands to Shift-F key combinations by selecting the Shift option. This means that you can assign up to 30 commands and scripts. Simply press the Shift key and the function key in order to use this second group of function key assignments.

You can see your first group of F-keys (without the modifying Shift key) on the FKeys palette, described on page 26.
7.3.22 Preferences

The Preferences dialog gives you control over some of IPLab's default behaviors. Note that several of these preferences only take effect for new images created after the preferences were set.

The Three Tabs of the Preferences Dialog Box

General Tab: The Default File Format governs which file format will be selected in the Save As dialog. This default file format will be used for all new images for which it is appropriate. (For example, EPR-format images must contain Unsigned 16 data. Save As will only default to the EPR format if the data is of type Unsigned 16; otherwise, it will default to IPLab format.)

The date format is used everywhere that IPLab displays a date as text (for example, in text annotations and in the base portion of indexed names).

Images Tab: The default normalization mode (the Based On setting in the Normalization command) will be used for all new images.

By setting Default Normalization to Custom, you can set the Min and Max values used by the Normalization command. The Sample Bounds menu at the bottom provides a shortcut for entering the numeric ranges associated with common data sizes (8 bit, 10 bit, 12 bit, etc.).

Drawings Tab: Measurement commands can label measured objects with numbers. This tab lets you control the formatting of those numeric labels. The radio buttons under Label Position let you control where IPLab will place the number in relation to the object's centroid. The Font, Size, and Style controls format the number's appearance.
7.4 View Menu

7.4.1 Show/Hide Image

The Show Image command causes the front window’s data to be visible; the Hide Image command conceals the data by filling the window with the word “Hidden”. Image processing is faster when the image is hidden, because the computer does not spend time updating the image’s display. The command’s name changes depending on whether the front window is already shown or hidden.

When new images are acquired or made by a script, they are automatically hidden in order to speed up the script. You can script the Show/Hide Image command to make display images or to hide them again. When you include this command in the script, the following dialog box will appear:

![Scripted Show/Hide Image Dialog Box]

7.4.2 Home Image

The Home Image command makes the upper left corner of the image visible in the upper-left corner of the window. This is very useful when you have magnified, demagnified, or dragged the image and now want to reset the image’s position. The Home key on the keyboard does the same thing.

7.4.3 Select Frame

Use this command to display a specific frame in an image sequence’s window. This is very useful for controlling image sequences from scripts.

![Select Frame Dialog Box]

Use the front window or pick an open window by name. When scripting, you can type a window name in the text box.

Select the Frame radio button in order to select a specific frame by its number. You can type the frame number or toggle the # to Var and use a variable. Remember that the first frame is always #0. The front window’s range of frames is displayed to the right of the Frame field.
Select the Previous or Next Frame radio buttons to move numerically down or up one frame. If you were at frame 0, Previous Frame would select the last frame (frame 2, in the above example), while Next Frame would select frame 1.

You can also switch frames within a sequence by using the Animate command or the keyboard. To use the keyboard, hold down the Command key (which may have an apple or a propeller symbol), and then press the up or down arrow to go to the next or previous frame, respectively.

### 7.4.4 Show/Hide ROI

When the ROI is showing, you will see its winking, thin, dashed outline. This command shows the dashed line if it is currently hidden, and hides it if it is currently showing. The command’s name changes depending on whether the front ROI is already shown or hidden.

Show/Hide ROI does not affect the location of the ROI. Also, all commands which operate on the ROI work the same whether the ROI is shown or hidden. The main reason for using this command is to view the data without the distraction of the blinking ROI outline.

The Show/Hide ROI command is scriptable; within scripts, it has the following dialog box:

![Scripted Show/Hide ROI Dialog Box](image)

### 7.4.5 Show/Hide Drawing, Show/Hide Segment

Use these commands to temporarily hide or re-display the overlay drawings and segments. The command names change depending on whether the drawings or segments are already shown or hidden. The overlays are otherwise unaffected.

These commands are scriptable, and present the following dialogs when in a script:

![Scripted Show/Hide Drawings and Show/Hide Segments Dialog Boxes](image)
7.4.6 Show/Hide Labels

Labels are the numbers that mark measured segments. You can choose to add labels or not in the Measurement Options dialog (Analyze menu). This command hides or re-displays the labels; it never changes or deletes the labels.

Show/Hide Labels is scriptable. When scripting, the following dialog box opens:

![Show/Hide Drawing Labels Dialog Box]

Scripted Show/Hide Labels Dialog Box

7.4.7 Photo

The Photo command displays the active image on a black background, without other windows, a cursor, or a menu bar to distract the eye. Once in photo mode, you can scroll the image anywhere on the screen by clicking the mouse button and dragging. The scroll mode cursor (the hand) will appear only when the mouse is being moved. You can exit the photo mode by hitting any key.

7.4.8 Animate

Use this command to simulate a movie by rapidly cycling through and displaying all of the frames in an image sequence. The image sequence must be the front, active window; this command will be grayed out for single-frame images.

If you have a video tape recorder attached to your monitor, then you can use the Animate command to present a movie to be recorded.

Animate works on image sequences of any data type. Animation of non-Byte type images may be slowed by the display normalization that must take place first. In this step, IPLab must find the minimum and maximum pixel values within each frame. You can help speed up the animation by first using the Normalization command with the Sequence, Saturated Sequence, or Manual settings. These normalization options reduce the time it takes to do normalization by eliminating the step to find the minimum and maximum pixel values in each frame.
Animate Image: You can display the animation inside the active window or in the center of the full screen. If you choose the **In Window** option, the **Status** palette will show you the animation speed in frames per second, right below the **Magnification** field. If you choose **Full Screen**, the animation will be displayed on a completely black background.

Animate Direction: The **Forward** option makes the animation run from the first to the last frame; the **Reverse** option makes the animation run backwards from the last frame to the first. The **Oscillate** option will make the animation run forwards and then backwards, giving a more natural feel to the display.

Display Sequence: If you select **Until User Stops**, then the animation will continue until the user presses any keyboard key. The second option lets you enter a specific number of times for the animation to run. This number can come from an **IPLab** variable. Displaying for a certain number of times is useful in a script in which you may want to animate the image sequence for a few seconds and then move on to some other operations.

Initial Display Rate: This lets you control how fast the frames will be changed. A rate of ten frames per second, for example, would mean that each frame is visible for 0.1 seconds before the next one is shown. This rate can come from an **IPLab** variable, as shown in the example dialog, above.

During the animation, you can step through the image sequence one frame at a time by hitting the left or right arrow keys. Then hit either the up or down arrow keys to continue the movie. If you are single-stepping through the image sequence when you stop the animation, then the frame you are looking at will still be the current frame. You can also use the up and down arrow keys to adjust the display rate.

Stop the animation by pressing any key on the keyboard besides the arrow keys.

7.4.9 View Options

The **View Options** commands let you control the display of different types of windows. They are all scriptable. Each of these commands affects only the currently active image.

The **View Options** settings will be saved with the image when you save it in the **IPLab** file format, regardless of which view is in use at the time.

7.4.9.1 Image View Options

This command lets you use dithering to improve the display of Color 24 and Color 48 images. The choice of whether to use dithering or not can be made for each image window separately. Newly created and opened images are, by default, not dithered. The **Dither** setting has no effect if your monitor’s color depth is set to millions of colors.

![View Options Image Dialog Box](image-url)
With **Dither Off**, each pixel is displayed in the color from the current CLUT which most closely resembles the true color of the pixel. Of course, the displayed color may be different than the true color if the CLUT does not contain the true color. The color mismatching that occurs in such cases can make your images look contoured.

With **Dither On**, such display errors are taken into account when the image is displayed. Dithering tries to adjust the displayed colors of adjacent pixels to compensate for the display errors. Dithering often provides a better looking image by fooling the eye into thinking that more colors are displayed than are really available. However, dithering comes at the price of slower redraw times, because the computer must calculate how to diffuse the color errors while it is redrawing the

### 7.4.9.2 Text View Options

**View Options: Text** allows you to specify the number of decimal places to use in displaying floating point data.

![View Options: Text Dialog Box](image)

### 7.4.9.3 QGraph View Options

**View Options: QGraph** lets you format the display of your QGraph. You can start out by viewing your graph and then using this command to make necessary changes to the display options. You can also access all of these dialogs by clicking either within the QGraph plot or upon one of the labels.

![View Options: QGraph Dialog Box](image)

The initial dialog, shown above, lets you select the plot type and the grid and background colors. The four buttons lead to four more dialog boxes for setting up the axes, the labels, and the data to be plotted.
When you click on the X Axis or Y Axis buttons, the following dialog box (or its Y equivalent) appears.

![X Axis Dialog Box for View Options: QGraph](image)

X Axis Dialog Box for View Options: QGraph

Use the Number Format section to display the axis numbers in scientific notation (or not) and with any number of decimal places.

The Tick Marks section controls how many numbered grid lines lie above or to the right of the edges of the graph. The number entered for the Use Delta option would be the distance between tick marks.

The Scale is the range of data that will be displayed on the axis. When you first start, check the Use Data Min/Max option.

The Use X/Y Axis Values button and the Make X/Y Axis the Same checkbox let you display the two axes the same way, based on either the open dialog’s settings or the other dialog’s settings.

Next, you can click the Labels button in the main dialog box.

![Labels Dialog Box for View Options: QGraph](image)

Labels Dialog Box for View Options: QGraph

Click on the appropriate Labels radio button on the right, and then enter and format the label on the left. Click OK to have all of the labels you changed applied to the graph.
Finally, click the **Columns** button in the main dialog box in order to choose which columns of data will be graphed.

![Columns Dialog Box for View Options: QGraph](image)

**Columns Dialog Box for View Options: QGraph**

*IPLab* can graph ten columns of data in a single QGraph. The values in those data columns will be plotted along the Y axis. For each of the ordinates, numbered Y0 to Y9, identify the column and choose the line pattern, width, and color; and the data point token and token color.

The **Plot Against** options let you plot the data against a linear X axis or against another column of data. When graphing the results of a measurement, you may want to set **Plot Against** to column 0, which holds the measurement numbers.

### 7.4.9.4 Perspective View Options

This dialog lets you set the viewing angle and other options for the three-dimensional perspective view. Each image window has a different set of perspective view options. This command is scriptable.

![Perspective View Options Dialog Box](image)

**Perspective View Options Dialog Box**

Using each of the checkbox options results in the best looking display, but takes the most time to draw. Removing hidden lines takes the most time, so you may want to experiment with different angle views without removing hidden lines, and then for the final display re-select the **Remove Hidden Lines** option.
7.4.10 View As

This hierarchical menu gives you four choices for viewing the data in the active window as an image, as text, as a QGraph, or as a three-dimensional perspective drawing. The same data (or a subset thereof) is being displayed in all views. This command does not change the data or even create a new window. You can control the displays of each view by using the View Options command, described immediately above.

When working with 2D data (with more than one column or row of data), View As: Image creates a two-dimensional display of the data, where each datum is shown as a gray or colored pixel. This is the normal image view with which you are familiar. When working with 1D data (having only one column or row of data), View As: Image plots the data along a curve. You can see an example of 1D data on page 10.

View As: Text displays the numerical value of each datum in an array. Because this looks like a spreadsheet, we sometimes call this a table. It is helpful to think of the text view as being identical to the image view, only without the numbers being translated into colored dots. For color images, the red, green, and blue values for each pixel will be displayed in each of the array’s cells. Text views are especially useful when viewing measurement results or performing arithmetical calculations. In order to select a range of data in a text view, click on one cell (that is, the pixel), hold down the Shift key, and then click on another datum.

7.4.10.1 QGraph View

QGraph is a special plotting utility which provides several options for creating high quality, annotated graphs of the columns in your data.

When data is displayed as QGraph, several pop-up menus allow you to customize the look of the graph. You can change the main graph and each of the labels by clicking and holding down the mouse over the main plot or any of the labels. These pop-up menus bring up the same controls as the View Options: QGraph command; you can read about those controls in the section immediately preceding this one. The QGraph window can be manually sized by clicking in the lower right corner of the window and dragging.

If you wish to use the plot as a picture in another application, you can transfer the plot via the Macintosh Clipboard. If you perform a Copy (from the Edit menu) when viewing as QGraph, you can paste the result into the Macintosh Scrapbook or some other application, such as a word processor or page layout program. However, the image paste buffer within IPLab will be unaffected, meaning that selecting Paste within IPLab will not produce the QGraph or its data. You can also use the Export: View to File or Export: View to Clipboard commands to copy and save the QGraph view.

QGraph views will always be printed at 72 dpi, and will always be scaled to fill an entire single page. Also, the entire ROI will always be printed.
7.4.10.2 **Perspective View**

*View As: Perspective* plots the image data that is within the ROI as a surface in 3D space, using the intensity as the z value. Since only the data within the ROI is represented in the perspective view, you must be certain to select the portion of the image that you want to see.

You can alter the parameters used to draw the perspective plot using the *View Options: Perspective* command. You can enlarge or shrink the perspective view window by clicking and dragging the grow box found in the lower right corner of the window.

Since the update of the perspective view’s plot is relatively slow, and since you may not want to wait for the entire update to occur, you can suspend the update by clicking the mouse within the perspective view plot window. You can force a new update later by resizing the window or by clicking the Collapse box in the window’s upper right corner.

If you wish to use the perspective plot as a picture in another application, you can transfer the plot via the Macintosh Clipboard. If you perform a *Copy* (from the *Edit* menu) when viewing as perspective, you can paste the result into the Macintosh Scrapbook or some other application, such as a word processor or page layout program. However, the image paste buffer within *IPLab* will be unaffected, meaning that selecting *Paste* within *IPLab* will not produce the perspective plot or its data. You can also use the *Export: View to File* or *Export: View to Clipboard* commands to copy and save the perspective view.

Perspective views will always be printed at 72 dpi, and will always be scaled to fill an entire single page. Also, the entire ROI will always be printed.
7.5 Enhance Menu

These commands provide built-in convenience for improving the appearance of your images. Many of the commands in the Enhance menu give you the option of creating a new window to hold the results of the processing you are performing. This gives you an opportunity to test the operation without destroying your original data. Creating new windows also gives you an “audit trail” of operations have performed to enhance the image.

7.5.1 Normalization

The Normalization command improves the display of the image without changing the data by controlling the mapping, or pairing, of the data values to display values. This provides you with two entities: the data, on which the program performs measurements, and the display of the data, which you visually judge.

Normalization will work on the active image so long as it is not pseudocolored with anything besides a monochrome CLUT, such as Monochrome, Red, Green, Yellow, etc. (If the Normalization command is grayed out, choose the Pseudocolor command and pick the Monochrome color table.) Each window has its own normalization parameters, so you can adjust the view of each one independently. When you choose the command, the following dialog box will appear:

To normalize the active image, choose the mode that the display should be Based On. Then adjust the display using the Contrast, Brightness, and Gamma controls. You can also use the more advanced histogram section under the blue triangle.

**Channel:** When working with a color image, this pop-up box lets you normalize all three color channels at once, with All The Same; or normalize only one channel at a time: Red, Green, or Blue. Grayscale images only have the Grayscale channel, of course. When using All The Same, all of the channels will have the same minimum, maximum, and gamma values.
**Based On:**

The **Based On** options control where *IPLab* will search for the minimum and maximum data values, which will be assigned to the display values 0 and 255, respectively.

When the **Frame** or **Sequence** option is used, the minimum and maximum data values will be found within either the current frame or image sequence.

With the **Saturated Frame** and **Saturated Sequence** options, too, the minimum and maximum values will be found within the current frame or sequence; however, approximately 1% of the darkest and brightest pixels will be mapped to 0 and 255. This feature displays better-looking images in cases where there are just a few pixels with values far from the typical image pixel values. This might happen, for example, when you grab an image from a camera with a faulty sensor.

You use the **Manual** option whenever you use the **Contrast**, **Brightness**, and **Gamma** buttons or the histogram section at the bottom of the dialog box. To see that section, you should click on the blue triangle at the bottom of the dialog so that it points down.

**Use ROI Values:**

This option modifies the **Based On** setting. When you click this button, *IPLab* will search for the minimum and maximum data values only within the selected region of the current frame or image sequence. The **Based On** setting changes to **Manual**, and the dialog box expands to show the histogram controls (as shown in the picture above and on the right).

**Normalization** will only consider rectangular ROIs; it will use the bounding rectangle around non-rectangular ROIs.

**Contrast, Brightness, Gamma:**

These three sets of controls work in the same fashion: left arrows decrease the parameter, and right arrows increase the parameter; double arrows make bigger changes than single arrows.

The circles return **Contrast** and **Brightness** to the minimum and maximum values present in the image and **Gamma** to a value of one. Clicking any of these buttons changes **Based On** to **Manual** mode.

**Contrast:**

Increasing the contrast narrows the range defined by the minimum and maximum normalization values, improving the display of that range of values at the expense of the rest of the image.

**Brightness:**

Increasing the brightness increases the number of pixels being displayed as white (or as the saturated channel color). This occurs by decreasing the minimum and maximum normalization values equally.

**Gamma:**

The gamma controls how evenly the whole range of data values are normalized. Increasing the gamma increases the contrast between higher data values and decreases the contrast between lower data values.

Gamma correction is often used to correct for the effects of nonlinearities in the image acquisition or display process. Any gamma setting other than 1.00 creates a non-linear mapping of the form:

$$\text{Displayed value} = \left(\frac{\text{normalized value} - \text{min}}{\text{max} - \text{min}}\right)^{\text{gamma}}$$

**View Saturated Pixels:**

This option helps you spot the highest and lowest-valued pixels. Saturated pixels cannot provide much information if they are the minimum or maximum values the camera (or other data collector) can possibly collect. When this is checked, white pixels will be displayed as red, and black pixels will be displayed as blue.
Histogram Section: In order to see (or to hide) the histogram section of the dialog box, click the blue triangle located below the View Saturated Pixels checkbox. The controls in this section, shown in the picture on the right on page 207, give you advanced control over the display of the image.

Mid Point / Gamma: Gamma is a function of the mid point value. Mid point is expressed as a percentage of the range between minimum and maximum. The Mid Point control will immediately apply any whole number from 5 to 95.

Histogram: The large rectangle holds a column graph of pixel intensities (on the x axis) and the quantities of pixels representing each intensity (on the y axis). This histogram helps you visualize the relative amounts of intensities within the image.

The green line on the histogram represents the gamma curve.

Type: You can view a Regular, Log Regular, or Integrated Histogram.

From: The histogram’s information can come from the current frame or the entire sequence.

Min / Max: Below the histogram are controls for manually setting the minimum and maximum data values that will be matched to the display values 0 and 255.

You can type the minimum value in the left text box, and the maximum value in the right-hand text box. Tabbing between text boxes applies your changes. You can also set these values manually by dragging the blue and red handles back and forth across the histogram. Any pixel intensities at the blue bar or to its left will be displayed as zero, or black. Anything at the red bar or to its right will be displayed as 255, or the saturated channel color. Altering the Minimum or Maximum changes Based On to Manual mode.

7.5.1.1 How Normalization Works

Normalization maps the data, which can have millions of different values, to the display values. Display values for grayscale images range from 0 to 255; display values for color images fall between the RGB values (0,0,0) and (255,255,255). The display value 0 corresponds to black. The display value 255 corresponds to the saturated color: bright white, red, green, or blue. Normalization maps data values greater than or equal to the Maximum parameter into the display value 255, and data values less than or equal to the Minimum parameter into the display value zero.

Remember that Normalization does not affect the data element values themselves. Only the display values, i.e. the way you see the data, are altered.

7.5.1.2 Normalization: New and Old Commands

Users of older IPLab versions should note that Normalization takes the place of the old Normalization, Adjust Contrast, and Color Balance commands.
7.5.2 White Balance

The White Balance command adjusts the intensities on all of the color channels so that the pixels within the ROI become approximately gray. This option allows you to correct for small amounts of color shading which are sometimes found in images captured by uncalibrated color cameras.

Before using this command, select a region of the image that should be white or gray. When you choose the White Balance command, the following dialog box will appear:

![White Balance Dialog Box](image)

The ROI flashes in the dialog’s small version of your image. You can click on the ROI and drag it around this small image. Clicking the Apply button will white balance the image and leave the dialog box open. By dragging the ROI and using the Apply button, you can search for the best white or gray region with which to balance the image.

Clicking the Apply to All Frames checkbox lets you white balance all of the frames in an image sequence the same way. If this box is unchecked, only the current frame will be white balanced.

Clicking OK will apply the white balancing and close the dialog box. Clicking Cancel will close the dialog box without making any changes to the image.

**Note:** White Balance does change the data in the front window. Make certain to work on a copy of your original data, or save the altered data with a new file name.
7.5.3  Equalize Contrast

This command provides the best contrast enhancement for the image by computing the “optimum” transformation on the color table or the image data. The command uses the following procedure for equalizing the contrast:

1. Compute the histogram of the image.
2. Integrate the histogram to form the cumulative distribution.
3. Scale the cumulative distribution to a mapping which has values between 0 and 255.
4. Apply the resultant mapping to either the image data or the CLUT.

Before using this command, you may want to select the region you most wish to enhance. Choosing the Equalize Contrast command will produce this dialog box:

![Equalize Contrast Dialog Box](image)

**Data In:** You can enhance the contrast of the front window or any other open window.

**Get Statistics From:** These options allow you to specify whether to get the histogram from the entire image or just from the ROI. If you are interested in enhancing the contrast of a specific portion of the image, you should select that portion and click **Get Statistics From ROI** in the dialog. That way, only the data within the region of interest will be used to equalize the contrast.

**Change:** You may elect to apply the mapping to the CLUT only, which will affect the display but will not affect the data values at all. Instead, you may apply the mapping to change the data values themselves in the entire image or in the ROI you have selected. If you change the data values, then the CLUT will not be affected.

As a rule of thumb, you should change the CLUT if you want to maintain the data values for quantitative analysis. Use **Data in Entire Image** or **Data in ROI** if you want to make all or part of the image look better and you don’t care about specific data values. Those two options are particularly useful if you plan to export the image to another computer.

**Create New Window:** Use this if you don’t want the existing data or display to be altered.

**Equalize Contrast** can be applied only to images of type Byte, Short Integer, and Unsigned16.
7.5.4 Image Ratios

Image ratios are used in a variety of applications to correct or calibrate image data. *IPLab* provides a single command with options for three different ratio operations. The formula for each selected option is displayed below the radio buttons. You may apply the command to the active window or to any open image window.

### 7.5.4.1 Flat Field

The **Flat Field** option is generally used to correct for non-uniform lighting and defects or non-uniformities in an imaging detector array. Such non-uniformity is really a “multiplicative noise” problem, so it really requires a division operation to correct for it. This correction method is more accurate than simply subtracting the background, although background subtraction is often performed to correct for uneven illumination because it is so simple.

To perform flat fielding, you should capture three images:

1. A dark current image, *i.e.*, an image taken with the camera shutter closed;
2. A uniform field, *i.e.* an image of an evenly illuminated field (this might be a blank field without a specimen in the field of view, for example); and
3. A data image, *i.e.* the image to be corrected.

Be sure all three images have been captured with the same integration time. In both Dark Window 1 and Dark Window 2 fields, select the single dark current image you have captured.

Although you get the best results if you have both the dark current image and the uniform field image, you can do some illumination correction with background subtraction only. If you do not have a uniform field image, you can just leave that name blank, in which case the denominator will be set to value 1.

The Flat Field formula is:

\[
\text{Result} = \frac{\text{Mean}(\text{Uniform} - \text{Dark}2)}{\text{Uniform} - \text{Dark}2} \times \frac{\text{Data} - \text{Dark}1}{\text{Data} - \text{Dark}1}
\]

The Flat Field option is performed “in place”; that is, it changes the values in the data window, and leaves the image in the same data type as the original.
7.5.4.2 **Ratio Only**

Use this option to produce a more accurate flat field result, or to calculate ratios of images taken at two different wavelengths, such as for fluorescence ratio imaging studies. Select the numerator image’s name in the **Data** field; select the denominator image’s name in the **Uniform** field.

When doing fluorescence ratio imaging, put the second wavelength image in the **Uniform** field. To get the best results, you should take two dark current images, one at each wavelength. If you don’t have the dark images, you can just leave those names blank.

The **Ratio Only** formula is:

\[
\text{Result} = \frac{\text{Data} - \text{Dark1}}{\text{Uniform} - \text{Dark2}}
\]

The **Ratio Only** option produces a new, Floating Point type image.

7.5.4.3 **Optical Density**

The **Optical Density** calculation is similar to the **Ratio Only** calculation, except that a base 10 logarithm is performed after the ratio is taken.

The **Optical Density** formula is:

\[
\text{Result} = \log_{10} \frac{\text{Uniform} - \text{Dark2}}{\text{Data} - \text{Dark1}}
\]

The **Optical Density** option produces a new, Floating Point type image.

7.5.5 **Invert**

Some image files brought from other programs may use the convention that 0 = white and 255 = black, unlike **IPLab**. This command lets you invert those images. You can also use this command to invert standard **IPLab** images to conform to the needs of those other programs. The dialog shows two items that may be inverted: the data and the CLUT.

![Invert Dialog Box](image)

**Invert** Dialog Box

If you invert only one of these items, the image will look different on your screen. If you invert both items, the image will look the same, but the data will be different. The formula used to invert the data and CLUT is:

- New value = 255 - old value for Byte type data and CLUT entries
- New value = old data max - old value for all other data types

To invert the colors (e.g. black to white), invert the CLUT; it may be wise to leave the data unchanged. When strong signals have been given low intensity values (e.g. some gel images), make quantification measurements possible by inverting the data values so that strong signals have high values.
7.5.6  Pseudocolor

The Pseudocolor command alters the colors used to display an image. You can do this to enhance the contrast between different elements of your image or just to color a grayscale image with the color of the filter used to capture it.

Each image window has its own Color Look Up Table (CLUT) assigned to it. The Pseudocolor command changes the CLUT associated with the active window. The dialog lets you choose from a pop-up list of Pre-Built CLUTs, the CLUT in the open window of your choice, or from mono-color CLUTs.

Please note that this command does not change the data; it only changes the display of the image.

Pseudocolor Dialog Box

Pre-Built: Use this pop-up list to choose from the standard CLUTs that are shipped with IPLab, as well as from the custom CLUTs which you create with the Edit Color Table command.

In Window: You may assign an arbitrary CLUT to the front image by supplying the name of an open Byte-type window that stores the color table in 256 rows and 1 or 3 columns. This information may be brought from another program or from within IPLab.

If you wish to use the CLUT from an IPLab image, open the image and then use the Extract: Color Table command (Analyze menu). This will create a 256x3 color table “image” named *.CLUT, which you should pick in the Pseudocolor dialog’s In Window field.

If the CLUT data window has a width of 3, then the first column (#0) will be used as the red component, the second column (#1) will be used as the green component, and the third column (#2) will be used as the blue component. If the CLUT data window has a width of 1, then the red, green, and blue components will be given the same values, producing a grayscale CLUT.

When scripting: You may enter any window name for the custom CLUT, even if the named window does not exist at the time you are creating the script. This allows the script to obtain the CLUT information from a data window that may be created earlier in the script.

Red, Green, Blue… These six radio buttons let you apply a monochrome CLUT that runs from pure black to a single, pure, color, as an alternative to the standard monochrome grayscale.

Apply: This applies the selected CLUT, so that you can immediately observe the effect.

OK, Cancel: The selected CLUT will be applied when click OK. If you click Cancel, nothing will be changed.
7.5.7 Edit Color Table

With this command, you can create your own custom Color Look Up Table (CLUT) and see it applied to your image in real time. CLUTs saved by the Edit Color Table command will appear in the pop-up list of CLUTs in the Pseudocolor command. Like the Pseudocolor command, this only changes the display of the image, and does not change the data.

![Edit Color Table Dialog Box](image)

The dialog shows a color bar at the top so you can see the CLUT you are creating. Click in the space below the color bar to flag points in the color table. Select flags by clicking on them. Selected flags have solid drop lines, while deselected ones have dotted drop lines. You can drag the selected flag to a new position on the CLUT. You can also delete it by pressing the Delete key on the keyboard.

Click the Histogram checkbox to display the image’s histogram across the flag area. Click the Revert button to remove all recent all flags from the dialog. The cursor’s location on the CLUT is shown below the flag section; in the example dialog, the cursor is at 65. The range affected by the selected flag is displayed below that; in the example dialog, the selected solid flag is affecting CLUT values 65 through 110.

The Solid, Ramp, Single, and Original radio buttons define what happens to the right of each flag. When the Original radio button, the default setting, is selected, that point in the CLUT will have the original value it had in the CLUT being edited. Click the Solid radio button to make all entries in the CLUT to the right of the flag a single, solid color. Click the Ramp radio button to create a ramping blend of colors between the selected flag and the next one. If you select the Single option, then only the CLUT entry at the flag will have the color you pick.

To get advanced control over which color table you edit, click the Show List button (which will become Hide List). A list of pre-made color tables and four buttons will appear. Select one of the pre-made color tables in order to edit it. Click the Save As button to save your color table under a unique name. (If you’re editing a bulleted list item, you’ll need to use Save As and give it a new name.) Click the New button to create a new color table (based on the previous one) to be edited. Preserve it for later use with the Save button. Use the New and Save buttons to make one color table after another. Dispose of a color table by selecting it in the list and clicking the Delete button.

Clicking OK will close the dialog and apply your newly edited color table to the image. Clicking Cancel will close the dialog without making any changes to the image.
7.5.8  Apply Color Table

The Normalization, Pseudocolor, and Edit Color Table commands create and attach various color tables to an image, but they do not change the data values of the image. The Apply Color Table command replaces the data value for each pixel with the pixel’s display value(s). The original data will be gone, but you can now alter the new data in order to change the display. This command only works on images of type Byte.

If the image had been pseudocolored with some form of a monochrome CLUT (only black, white, and/or grays), then the new image will be a monochrome Byte image. Pixels that had been pseudocolored white (255), for example, would be given the intensity value 255, whatever their original value. Meanwhile, the CLUT would become the standard, built-in Monochrome CLUT. The result is that the image would look the same, but all of the data would be changed to reflect the old CLUT.

If the image had been pseudocolored with a more colorful CLUT (having anything else beside black, white, and gray), then the new image will be a Color 24 image which looks exactly like the original Byte image. Each pixel in the image would be replaced with a 24 bit color pixel whose red, green and blue values come directly from the CLUT entry for the value of that Byte pixel. Pixels that had been pseudocolored with a shade of red (255 15 10), for example, would become a Color 24 pixel with a red intensity value of 255, a green intensity of 15, and a blue intensity of 10.

Note: Apply Color Table does change the data in the front window. You should save your data before using this command.

7.5.9  Color To Grayscale

This command operates only on Color 24 and Color 48 images. It creates a new image window which is the grayscale equivalent of the color image data. The formula used is the same one that is used to split the luminance (Y) channel from a color image using the YIQ color coordinate system. This is the way you would see a color picture on a black and white television.

7.5.10  Despeckle

Use this command to replace pixels that differ significantly from their surrounding neighbors. This command helps remove spurious pixels caused by minor defects in CCD chips, or other acquisition hardware.

Only pixels within the ROI are changed. The replacement criterion is based on the difference between the value of each pixel and the median of it and its neighbors. For pixels on the edge of the image, only those neighbors within the image are tested. The replacement value is the median of the pixel and its surrounding neighbors.

Despeckle Dialog Box

The top two radio buttons let you choose which window to use. You can use the active window, or you can pick the window by name.
If you are scripting the command, then you can type the name into the **Window** text box before the window is open.

The edit box lets you enter the percentage by which a pixel value must exceed the median of its neighbors to be replaced.

When the **Create New Window** checkbox is checked, the operation will be performed on a duplicate of the given window, leaving the original window unchanged.

Check the **Do All Frames** checkbox to despeckle an entire image sequence.

### 7.5.11 Linear Filter

In linear filtering, each pixel in the ROI of an image is changed based on the values of nearby pixels. Use this command to apply a pre-defined linear filter to the pixels within the ROI of any open window.

You can get the filter definition in one of two ways: (1) from another window, or (2) from one of the built-in filters. Check the **Do All Frames** checkbox to filter an entire image sequence, instead of only the current frame.

If you choose the **Use Filter in Window** option, you can apply an arbitrarily sized image kernel; however, this kind of filtering operates much slower than the built-in filters. The window containing the filter kernel must be either Long Integer or Floating Point data type. Pre-defined filter kernels are supplied with in the **Kernels** folder within the **IPLab** folder. You can use these as examples for making your own kernels. The ROI on the original image will be reduced in size, if necessary, to eliminate edge effects. A new image will be created to hold the filtered results.

The **Divide by Filter Sum** checkbox lets you specify whether to divide the filtered image by the sum of all of the filter elements. If the filter data window is Long Integer, you should probably have this box checked (the default). If the filter data window is Floating Point, you can scale the kernel so that the sum of all of the elements is 1.0. If the sum of filter elements is 0 and this box is checked, an error will be generated and the filter will not performed.

Click the **Use Built-In Filter** option to pick from the list of filters shipped with **IPLab** or created using the **Define Linear Filter** command (Enhance menu). The built-in filters are limited to 5x5 kernel sizes, but they operate faster and they give you more options (see the description of **Define Linear Filter**). If you do not want the original data to be altered, click the **Put Results in New Window** checkbox. Otherwise, make sure you save the data first.
7.5.12 Median Filter

The median filter acts like the linear filter, in that each pixel in the ROI is changed based on the values of nearby pixels. In the case of the median filter, the pixel value is changed to have the median value of the pixels within the neighborhood, i.e. half the pixels in the neighborhood will have values above the median and half will have values below the median. The size of the neighborhood around each pixel used in determining the new pixel value is set by clicking within the gray box labeled Mask.

![Median Filter Dialog Box]

Median Filter Dialog Box

The median filter is often used to remove salt-and-pepper noise in images. It eliminates isolated bright pixels which are due to cosmic rays hitting the detector of your camera, for example. Clicking within the gray Mask box sets the size and shape of the neighborhood of pixels used in determining the new pixel value. The small squares within the Mask box represent pixels beneath the mask. The color groupings show you how large a neighborhood you can choose. In the dialog above, a 3x3 mask has been chosen. The mask is always centered over the pixel being filtered. Normally, you will use a small mask size, such as 3x1 or 3x3. If your image has a lot of noise, you may want to experiment with 3x5 or 5x5 masks.

Since this command changes the data, you should click the Create New Window checkbox.

Check the Do All Frames checkbox to filter an entire image sequence, instead of only the current frame.

7.5.13 Edge Filter

This command lets you apply one of several built-in, nonlinear filters which are designed to detect edges in an image. The edge filters change the value of each pixel in the ROI based on the values of nearby pixels.

![Edge Filter Dialog Box]

Edge Filter Dialog Box

Edge Filter may be applied to the active window or any open window you pick. In this way, you can process images without bringing them to the front, and you can script this command without the window being open yet.
The **Roberts** edge filter applies the following 2x2 linear filter kernels to the original image and sums the absolute value of the results for each pixel:

\[
\begin{bmatrix}
-1 & 0 \\
0 & 1
\end{bmatrix}
\quad\begin{bmatrix}
0 & -1 \\
1 & 0
\end{bmatrix}
\]

The **Sobel** edge filter applies the following 3x3 linear filter kernels to the original image and sums the absolute value of the results for each pixel:

\[
\begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\quad\begin{bmatrix}
-1 & 0 & 1 \\
0 & 2 & 0 \\
2 & 0 & 2
\end{bmatrix}
\]

The **Sobel** filter is a little less noise-sensitive than the **Roberts** filter, because the **Sobel** filter takes into account the values of more neighborhood pixels.

The **Morphological Gradient** computes the difference between the original image and a 3x3 eroded version of the image. This is an excellent and fast alternative to the other edge filters.

Click the **Create New Window** checkbox if you have not already saved the image. That way, you can see the edges in a new window without altering the original data.

Check the **Do All Frames** checkbox to filter an entire image sequence, instead of only the current frame.

### 7.5.14 FFT Filter

After taking the Fourier transform of an image, you can use this command to apply Fourier domain filtering techniques. The **Transforms** command (**Analyze** menu) creates two windows which are either the Real and Imaginary or the Magnitude and Phase representations of the Fourier transform. You must enter in this dialog the names of both the Real and Imaginary windows or the name of the Magnitude window. In the former case, both windows are operated on in the same way. In the later case, only the magnitude needs to be processed.

![FFT Filter Dialog Box](image)

**FFT Filter** Dialog Box

A common use of FFT filtering is to eliminate periodic noise. Below is the magnitude FFT of an image that was captured from a composite color camera with a grayscale frame grabber. The image has periodic noise due to the chroma signal in the composite image. This periodic noise appears as bright points away from the DC term in the magnitude FFT.
Eliminating Periodic Noise

The key to filtering out noise spikes in the Fourier domain is to apply the masks symmetrically about the DC term. To do this, you use one of the ROI tools to outline one of the noise areas, and apply the FFT filter command with the Clear Inside ROI Symmetrically operation. The result is shown next to the original FFT. In this case, you would perform the same operation on the noise terms in the other quadrants of the FFT. You would then take the inverse transform to see the results in the image plane.

You can also use this command to specify Low Pass, High Pass and Band Pass filters as a percentage of the full spectrum. Below left is an example of a Low Pass filter with a 60% High Cutoff.

It is also possible to define an almost arbitrary Low Pass filter. First, use the freehand ROI tool to outline roughly the region you want to pass. Then apply this command with the Clear Outside ROI Symmetrically operation. This makes the region properly symmetric before clearing.
Mathematical morphology provides four nonlinear filters that help enhance and analyze images.

These filters change each pixel in the ROI based on the minimum or maximum value in the local neighborhood around the target pixel. The size and shape of that neighborhood is set using the Mask box.

**Erode:** The Erode filter computes the minimum value of the pixels within the neighborhood and assigns it to the target pixel.

**Dilate:** The Dilate filter computes the maximum value of the pixels within the neighborhood and assigns it to the target pixel.

**Open:** The Open filter performs an erosion over the whole image, and then a dilation.

**Close:** The Erode filter performs a dilation over the whole image, and then an erosion.

**Mask:** Clicking within the gray Mask box sets the size and shape of the neighborhood of pixels used in determining the new pixel value. The small squares within the Mask box represent pixels beneath the mask. The color groupings show you how large a neighborhood you can choose. In the dialog above, a 3x3 mask has been chosen. The mask is always centered over the pixel being filtered.

This command is most often used on images with values 0 and 1 or 0 and 255, but you can also apply the command to grayscale images.

Fill the Create New Window checkbox to perform the operation on a duplicate of the active window, leaving the original window unchanged.

Check the Do All Frames checkbox to operate on an entire image sequence.
7.5.16 Define Linear Filter

Use this command to create and edit named filters which are used by the Linear Filter command (Enhance menu). Linear filtering replaces the value of each data element in the ROI with a value computed by a weighted sum of the values of neighboring data elements in the input, and perhaps the values of some data elements already computed in the output. Although only data elements within the ROI are affected, some data elements outside the ROI are used in the computation for elements near the edge of the ROI.

Define Linear Filter Dialog Box

There are four parts to each filter:
- **Moving Average Component:** This kernel is applied to the input.
- **Autoregressive Component:** This kernel is applied to the output.
- **Divide By** parameter
- **Post Filter Offset** parameter

Enter positive or negative integers in the four parts by clicking in the boxes. You can move quickly to the next box by pressing the Tab key.

**Note:** All the parameters in the Define Linear Filter dialog must be integer values.

The Clear All button in the dialog allows you to quickly zero out all coefficients in the displayed kernel so you can start over. The buttons down the middle of the Define Linear Filter dialog let you edit the list of filter kernels. The New Kernel button brings up a dialog which prompts you for a new filter name and then presents you with a clear kernel. If you edit a kernel and make a mistake, you can use the Revert button to bring back the original kernel. Use the Save As button to save the displayed filter kernel with a special name, and use the Save button to update a filter kernel you have edited. The Delete button deletes the highlighted filter name from the list of filters. Click Done when you are done editing linear filters.

Any new kernels that you define are stored in the User Data file when you quit IPLab. See the section called Saving IPLab Settings in the Operation chapter of this manual.

**Note:** Define Linear Filter is not scriptable.

The two basic types of filters are Finite and Infinite Impulse Response Filters. They are described below. Edge effects are also described below.
7.5.16.1 Finite Impulse Response Filters

Finite Impulse Response (FIR) filters use only a weighted sum of the neighboring data in the input image to produce each output pixel. These weight values are entered in the **Moving Average Component** section of the dialog. If you label the moving average coefficients as follows:

\[
\begin{array}{ccccccc}
\alpha_{-2,-2} & \alpha_{-1,-2} & \alpha_{0,-2} & \alpha_{1,-2} & \alpha_{2,-2} \\
\alpha_{-2,-1} & \alpha_{-1,-1} & \alpha_{0,-1} & \alpha_{1,-1} & \alpha_{2,-1} \\
\alpha_{-2,0} & \alpha_{-1,0} & \alpha_{0,0} & \alpha_{1,0} & \alpha_{2,0} \\
\alpha_{-2,1} & \alpha_{-1,1} & \alpha_{0,1} & \alpha_{1,1} & \alpha_{2,1} \\
\alpha_{-2,2} & \alpha_{-1,2} & \alpha_{0,2} & \alpha_{1,2} & \alpha_{2,2} \\
\end{array}
\]

where \(\alpha_{0,0}\) represents the center value of the moving average kernel, then for an input image \(X\), the output \(Y\) at location \((h,v)\) is given by

\[
Y_{h,v} = \frac{1}{\text{Divide By parameter}} \sum_{j=-2}^{2} \sum_{i=-2}^{2} \alpha_{i,j} X_{h+i,v+j} + \text{(Post Filter Offset)}
\]

After applying this formula, the result is clipped so that the resulting values fall in the range of values allowed by the data type of the image:

- **Byte Type**: \((0, 255)\)
- **Integer Type**: \((-32768, 32767)\)
- **Unsigned16 Type**: \((0, 65535)\)
- **Long Integer Type**: \((-2^{31}, 2^{31} - 1)\)

7.5.16.2 Infinite Impulse Response Filters

Infinite Impulse Response (IIR) filters use a weighted sum of the data values already computed in the output image to produce each output pixel. These weight values are entered in the **Autoregressive Component** section of the dialog. Suppose we label the autoregressive coefficients as:

\[
\begin{array}{ccccccc}
\beta_{-2,-2} & \beta_{-1,-2} & \beta_{0,-2} & \beta_{1,-2} & \beta_{2,-2} \\
\beta_{-2,-1} & \beta_{-1,-1} & \beta_{0,-1} & \beta_{1,-1} & \beta_{2,-1} \\
\beta_{-2,0} & \beta_{-1,0} & \beta_{0,0} & \beta_{1,0} & \beta_{2,0} \\
\end{array}
\]

Then, given an input image \(X\), the output \(Y\) at location \((h,v)\) is given by:

\[
Y_{h,v} = \frac{1}{\text{Divide By parameter}} \sum_{j=-2}^{2} \sum_{i=-2}^{2} \beta_{i,j} Y_{h+i,v+j}
\]

\[
+ \frac{1}{\text{Divide By parameter}} \sum_{j=-2}^{2} \sum_{i=-2}^{0} \beta_{i,j} X_{h+i,v+j} + \text{(Post Filter Offset)}
\]

where \(\beta_{0,0} = \beta_{1,0} = \beta_{2,0} = 0\). For nonzero moving average coefficients, the two sums above are simply added.

With a combination of autoregressive and moving average components, quite complex filter characteristics can be generated. (Cascaded ARMA filters of small order can approximate a desired filter characteristic arbitrarily well.)

Note that when filtering 1D data, only the center rows of the moving average and autoregressive component matrices should be entered. The other components of the filter parts are ignored.
7.5.16.3 Edge Effects

You should understand the edge effects caused by filtering at the edges of the region of interest. At the edges of the ROI, the filter uses adjacent pixels in the input image that are outside the ROI when required to compute the output values. That occurs when the ROI is embedded in the middle of an image. However, if the ROI boundaries are at the boundaries of the image, then the image will be, in effect, treated as if it were embedded in a background of zero-value pixels. When values outside the image are required to perform the filtering, *IPLab* treats those values as if they were zero.

7.5.17 Flip

This command provides you with several options for geometric transformation of the data in the active window.

![Flip Dialog Box](image)

The picture in the dialog indicates what will happen when you select one of these options. Transpose flips the image over a diagonal axis, as shown above. Flip Vertical flips the image in the up-down direction, over a horizontal axis. Likewise, Flip Horizontal reverses the direction of the image from left to right. Turn 90°, 180°, and 270° all rotate the image in the counter-clockwise direction.

If the Create New Window checkbox is checked, the result will be placed in a new window.

This operation is reversible since no interpolation is performed. The result is that you may perform multiple flips without any cumulative changes to the pixel values.

You may apply any geometric operation to 1D data as well as to 2D data. However, you may not see any effect in some cases. For example, doing a Flip Horizontal on a 1D column of data has no effect. To transform 1D column data into 1D row data, use the Transpose option.
7.5.18 Register

The **Register** command performs geometric rotation, shifting, and scaling or one image to make it line up with another image. The idea is to align two images so that other commands can be used to compare them, for example, by doing image subtraction or image addition.

![Register Dialog Box]

**Register Dialog Box**

**Reference Window:**
If both the active image window and another (reference) window have registration marks, you can use this command to line up the registration marks on the two images. To do this, the reference window and the active window must have the same number of registration marks. Choose an open window (not the front one) from the **Reference Window** pop-up list.

**Calculate From…**
When the **Calculate From ® Marks** button is clicked, **IPLab** automatically computes the ΔX, ΔY, Angle, and Scale parameters which will bring the registration marks into alignment as closely as possible. It does this by comparing registration marks in the active window with those in the reference window.

**ΔX, ΔY,**
**Angle, Scale:**
These text fields hold the amounts of shifting, rotation, and scaling that will be applied to the front image. You may enter decimal (floating point) values for each of the parameters or have them calculated for you by using the **Calculate From ® Marks** button, described above. If the # buttons are toggled to **Var** (as with **Angle** in the above figure), then the values for those fields will come from the **IPLab Variables** array. (Please see the Variables section on page 119 for more detail on using variables.)

**Method:**
These are two interpolation methods for calculating the new, registered image. The **Nearest Neighbor** method is faster, while the **Bilinear Interpolation** method often gives a visually more pleasing result, because it smoothes jagged edges. Recent Macintoshes (G3s and later) are fast enough to let you use **Bilinear Interpolation** whenever you like. Generally, neither of these methods can be reversed by doing the “inverse” operation.

**Create New Window:**
Click this checkbox to put the registered results in a new window of the same size. If you do not use this option, make sure to save your data before registering, because the result will be placed in the same window.

**OK, Cancel:**
When you are done, click **OK**, and the front window will be registered. Click **Cancel** to cancel the action.

If the image has no registration marks, it is rotated and scaled about the image center. If there are registration marks, the image is rotated and scaled about the centroid of the registration marks. To place registration marks, use the Registration tool (®) from the **Tools** palette, which is described in the **Operation** chapter of this manual.
After using **Register**, you can use the **Mosaic** command (**Enhance** menu) to join the two images. You can use the **Image Arithmetic** command (**Math** menu) to subtract similarities between the two images and look for changes. Finally, you can use the **Merge Color Channels** command (**Math** menu) to blend them into a full color image. See the **Flip** and **Rotate & Scale** commands (**Enhance** menu) for other options in rotating and scaling images.

### 7.5.18.1 Scripting the Registration Command

The **Register** command can be scripted. The dialog changes slightly to give you a checkbox option if you want to have **IPLab** automatically compute the parameters by matching registration marks. If you select this option, the parameters boxes will be disabled.
7.5.19  Rotate & Scale

This command performs fractional geometric rotation and scaling of the active (front) image.

Rotate & Scale Dialog Box

In order to change the image size, you may enter different values in the Scale X and Scale Y fields, or lock them together with the Same Values checkbox. If you want to scale the image without any rotation, enter 0 in the Rotation Angle field.

In order to rotate the image, enter the angle, in degrees, in the Rotation Angle field. If you want to rotate the image without any scaling, enter 1 for each of the Scale fields.

The image is rotated and scaled about the center of the window.

If you do not check the Create New Window checkbox, then the existing window will be replaced by another window which is big enough to hold the modified image. However, if you do click the checkbox, then the original window will be left alone, and a new window of the appropriate size will be created.

Whenever fractional scaling and rotation are performed, a method of interpolation must be used. You have two options: The Nearest Neighbor option is faster, while the Bilinear Interpolation method often gives a visually more pleasing result, because it smoothes jagged edges. In general, the interpolation operations are not reversible. You cannot retrieve the exact original image by performing the “inverse” operations.

See the Flip command and the Register command (both in the Enhance menu) for other options in rotating and scaling images.
Mosaic

The Mosaic command provides a convenient way to join data windows into one larger data window. The Mosaic command is also useful for joining measurements or histograms from the analysis of multiple images.

First, choose the two images to be joined; you can choose the front window or any open windows. The source windows need not be the same dimensions for any of the Placement options; however, both source windows must be of the same data type.

Next, pick one of the three Placement options for placing the two source windows in the new window. Horizontal creates a new, wide window with the two images attached side by side. Vertical creates a new, tall window with the first image above the second image.

Alternatively, you may use the Registration Marks option to attach the two images with their registration marks lined up. This command does not perform any rotation or scaling on either source image in lining up the registration marks. The centroid of the registration marks is aligned by simple shifting. You should use the Register command first to rotate, scale, and shift the images, and then join them with the Mosaic command.

Checking the Do All Frames box mosaics two sequences into one larger sequence, so long as both source windows have the same number of frames. Otherwise, the current frame of each image will be used to produce a single mosaic frame.

When the two images are being joined, the second image is drawn in the new window after the first. This has two consequences:

1) When using the Registration Marks option, the second image will be drawn on top of the first image wherever they overlap.

2) The CLUT of the second image will be used for the new image. If you wish to mosaic two images with different CLUTs and still preserve the colors, you should first use the Change Color Type command (Math menu) to make both images Color 24 or Color 48, and then use the Mosaic command.
7.6 Analyze Menu

7.6.1 Segmentation

The Segmentation command separates target pixels from background pixels based on their values, and places colored overlays (segments) on the target pixels. This marks the pixels (e.g. for measuring), but does not change the covered data.

The Segmentation command is organized around files called sets, which store the parameters for how to segment an image. When IPLab is launched, it searches the Segmentation Sets folder (found with in the IPLab folder), and loads all of the sets found, up to a limit of 32,000.

You can segment your image using a set by choosing one from the Set pop-up box and then clicking OK. When you first start out, the only set available will be the Default set. You must choose an existing set or create a new one by clicking Duplicate. You can then define what pixel values/ colors you want segmented. To do that, follow the instructions below:

![Segmentation Dialog Box, Showing the Simple Tab](image)

**Segmentation Color:** Click on this pop-up box to pick the segment color that this set will add.

**Eyedropper Grid:** Moving the pointer over the image will turn it into an eyedropper. Clicking on the image will pick up the pixel colors under the eyedropper and put them in the eyedropper grid. *Segmentation* keeps a list of what colors have been picked. The grid will always show the most recent item, if any, in this EyeDrop list. A grid element will display one of three things:

1. Checker Board pattern. This means that when the user clicks on an image, this element will not be defined.
2. An ‘X’. When the user clicks on an image, the color will fill this element.
3. A color. This is a defined element.

The slider controls the size of the eyedropper. The larger the grid, the more colors that can be picked up. The default eyedropper size is a 3x3 grid.

Press the Backward arrow button, to undo any manual changes made to the list of selected colors. Pressing the button again will remove the most recent item from the EyeDrop list; in effect, this lets you undo clicks of the eyedropper.

**Reset:** The Reset button sets the current set to the default parameters.
Duplicate: The **Duplicate** button creates a copy of the current set. All subsequent changes to the segmentation are saved as part of this new set. Sets are saved in the **Segmentation Sets** folder within the **IPLab** folder.

Delete: The **Delete** button deletes the segmentation set’s file from the disk and removes the set from the **Set** pop-up list.

Cancel: When you cancel out of the **Segmentation** dialog, changes made (but not saved) to segmentation sets will not be applied to the image nor saved.

Apply: The **Apply** button applies the current parameters to the image.

OK: Clicking **OK** applies the current segmentation set to the image and saves any changes made to sets.

All changes made to a set are immediately saved unless the **Cancel** button is pressed.

### 7.6.1.1 The Simple Tab

The **Simple** tab, shown on page 229, is the default tab for segmenting color images, and is only available for segmenting color images. The four sliders modify the eye-dropped colors in the R<sub>s</sub>, G<sub>s</sub>, B<sub>s</sub>, and Intensity channels. For a description of these channels, please see page 133 in the Data Types section.

Use these sliders before clicking the eyedropper on the image. The higher the sensitivity, the more closely the segments will match the colors you selected with the eyedropper. The lower the sensitivity, the more approximate the segmentation will be; you will segment colors that are close to the ones you eye-dropped.

Switching to this tab removes all manual changes and segments the image based upon the color list.
7.6.1.2 The Histogram Tab

The **Histogram** tab is available for both grayscale and color images. The **Histogram** tab lets you pick the range or ranges of pixel values to be segmented.

### Segmentation Dialog Box, Showing the Histogram Tab

A. The four radio buttons determine which channel the histogram will affect. The pop-up menus to the right of the radio buttons specify a Boolean operation between channels. To the right of the Boolean pop-up menus, the channel menus allow the user to select a channel to segment. The histogram will show the range selected for the current radio button.

You can use the four radio buttons to segment different ranges of values in the red, green, and blue channels, and thus segment a specific color. You can also segment different ranges of values within the same channel. Use the **AND** operation to select pixels that fall in both ranges of values. Use **OR** to select pixels that fall in one or the other range. Use **XOR** to select pixels that fall in one or the other, but not both. Choose **Off** to not use the next radio button.

B. The histogram itself displays the relative quantities of each pixel value. The color bar above the histogram represents the selected radio button’s channel. The slider to the histogram’s right raises the histogram up, allowing the user to see the smaller bars with greater ease. In addition, moving the cursor over the histogram changes it to a magnifying glass, letting you magnify the histogram by clicking on it. Hold down the Option key and click to zoom out again.

C. The range slider control is the area beneath the histogram. When the cursor moves over this area, it changes to a hand. Click the hand in this area to drag the min & max bars left or right without changing their relative distance.
Part of the control takes on the current segmentation color, indicating which part of the histogram will be segmented.

D. Select a range of pixel values by moving the min and max bars across the histogram. You can drag the bars by clicking on and dragging their handles; the min and max handles are blue and red, respectively.

E. You can also select a range of pixel values by typing the minimum and maximum limits in the two text boxes below the histogram. Type the min value in the left box and the max value in the right box. Tabbing from one text box to the other will apply your typed settings; you can also click the Apply button. The current channel’s minimum and maximum values are displayed below the text boxes.

F. The Invert button beneath the histogram, inverts the selection so that everything not between the min and max bars will be segmented.

7.6.1.3 The Color Cube Tab

The Color Cube tab provides a second method for segmenting color images. The color cube represents the color spectrum as a 3D graph, with the intensities for red, green, and blue plotted along the x, y, and z axes, respectively. Each color selected by the eyedropper represents a unique color in the standard color cube.

Segmentation Dialog Box, Showing the Color Cube Tab

The Sensitivity slider defines how much of a volume around each unique color should be considered the same color. Raising the sensitivity forces segmented colors to be closer to the colors you selected with the eyedropper. Lowering the sensitivity increases the number of similar colors that will be selected.

Checking Expand Selection… takes the volume defined by the sensitivity and multiplies it by the number in the pop-up menu.

The Drop Colors control lets you not segment colors represented by too many or too few pixels.

The black and white Invert button allows the user to invert the current segmentation defined by the parameters.
7.6.1.4 Segmentation: New and Old Commands

Users of previous versions of *IPLab* should note that this command combines the old Segmentation and Color Segmentation commands; the old Binarize Data function is now found in the Math menu’s Binarize command.

7.6.2 Segment at ROI

This command provides you with an easy way to place a segment outline where the ROI is. This is very useful when you want several hand-drawn segments that are exactly the same size. Use the following sequence of steps:

1. Draw the ROI using the rectangular ROI tool.
2. Apply Segment at ROI (or press the keyboard combination Command-2).
3. Hold down the Option key and click-drag the ROI to a new location.
4. Repeat steps 2 and 3 as necessary to make multiple segments.

7.6.3 Modify Segments

Modify Segments provides binary morphology operations for editing the segment layer of an image. Modifying the segment layer lets you make sure that the correct regions get measured and that the segments accurately represent the regions.

Modify Segments Dialog Box

The five Filter operations sequentially apply a mask (or structuring element) to each point in the image overlay. At each pixel, the 24 immediate neighbors of the pixel in the image overlay are compared to the mask, and the pixel is set either to the chosen segment color or to no segment (transparent) according to the following rules:

\[\begin{array}{ll}
\text{Filter Type:} & \text{Set Center to <Segment Color> If:} \\
\text{Erode:} & \text{all the green locations in the mask are segmented in the overlay around that pixel.} \\
\text{Dilate:} & \text{at least one green location in the mask is segmented in the overlay around that pixel.} \\
\text{Hit or Miss:} & \text{the green and white locations in the mask exactly match the segmented and empty (transparent) locations in the overlay around that pixel.}
\end{array}\]
The **Open** operation is an **Erode** followed by a **Dilate**, and the **Close** operation is a **Dilate** followed by an **Erode**, all using the same mask.

Click in the **Mask** box to change the color of each cell from white to green to gray. Gray indicates a “Don’t care” location, where the filter can act as if that location is either green or transparent; it is available only if you have the **Hit or Miss** option selected.

The last six options do not use a mask. **Complement** switches the green and transparent values over the entire overlay, so areas that were segmented are clear, and areas that were clear are segmented. **Thin** removes one layer of boundary pixels from each segment; **Thicken** adds a layer of boundary pixels to each segment. **Skeletonize** removes boundary pixels until only a linear path is left within each segment. **Fill Holes** fills in all of the holes within each segment. **Boundary Only** leaves only the outside boundary pixels in the segment layer. (If a segment has interior holes, the boundaries of those interior holes are not preserved by the **Boundary Only** option.)

### 7.6.3.1 Some Applications of Modify Segments

Among other uses, **Erode** is good for removing tiny segments created by the thresholding of background signal. **Open** is even better, since its **Erode** operation will remove most tiny segments and its **Dilate** operation will then replace the loss to the main segment’s edges. These actions are appropriate when the tiny segments should not be measured.

Likewise, **Dilate** is good for expanding a large segment to include tiny, particulate segments at its edges. The **Close** filter is better yet, since its **Dilate** operation will expand the large and tiny segments together, and then its **Erode** operation will remove excess segment from the edges.

You can use a combination of **Complement** and **Skeletonize** to delineate grain boundaries by the following steps:

1. Isolate the grains from the background with the **Segmentation** command. This produces a green overlay on the grains.
2. Use **Modify Segments: Complement** to set the background pixels to green in the overlay. Now the background is overlaid with green.
3. Apply **Skeletonize**. This produces green grain boundaries.

### 7.6.4 Change Segment Color

This command lets you conditionally change the color of segments.

![Change Segment Color Dialog Box](image)
At Pixels Where sets the condition for which pixels get the new segment color. The segment layer will change colors wherever this condition is met. If no segment layer exists on the image, one will be created.

- **Image Value Is:** All pixels with data values within this range will be given the new segment color.
- **Segment Color Is:** All pixels overlaid with this segment color will be changed.

The **To: New Segment Color** option sets the replacement segment color. Note that you can remove colored segments from specific sections of your image by choosing Transparent as the new color.

### 7.6.5 Set XY Units

Use this command to set the distance units. The **Predefined** pop-up list shows pre-defined units. To define your own units, use the **Define XY Units** command. If you don’t want to define your own units, you can enter the distance units in the **1 pixel** fields.

In addition to setting the units for the active window, this command sets the units for all new images acquired with a **Camera** menu command or created by the **New Image** command (**File** menu).

The **Set XY Units** dialog also lets you place a calibration bar in the image. This bar is drawn in the drawing overlay using the drawing color which is currently selected on the **Tools** palette.

![Set XY Units Dialog Box](image)

Once you have set the units for any window, all new windows created from that time on are assigned the same units, until you quit **IPLab**. This applies to windows created by the **New Image** command or by **Control** menu acquisition commands. When you save an image as an **IPLab** image, its units are saved with it. The default units when **IPLab** first starts are “1 to1 Pixels.”
Define XY Units

This command allows you to define distances within an image. Distances are measured in Euclidean space so that the distance between point \((a, b)\) and point \((x, y)\) is given by

\[
\text{Distance} = k \sqrt{(a - x)^2 + (b - y)^2}
\]

where \(k\) is a constant which scales the distance to the appropriate units.

For the units text string, you may enter any nine-character string. As an example, the units “Big Step” would be accepted. If you enter more than nine characters for units, a beep alerts you to the error.

IMPORTANT: If you define additional units in the Define XY Units dialog, it is important to keep distinct the strings you use for the units definition. For example, if you decide to keep the above units definition, you should not use the string “\(\mu (10x)\)” for any other units definition.

There are several options in how to define the numeric parameters in the dialog:

7.6.6.1 Interactively Defining Units

If you click on the Interactive button in the dialog, you will be instructed to then click on two points within the image. These two points must be a known distance apart. When you click on the first point, a line will be dragged from that point to the location of the cursor until you click on the second point. After you click on the second point, the “p” and “u” values in the formula

\[
p \text{ pixels} = u \text{ units}
\]

will be updated in the dialog. You should then set the name of the units and the real distance “u” between these two points in this formula.

7.6.6.2 Defining Units by a Known ROI Size

If you click on one of the ROI size buttons (ROI Area, Width, Height, or Perimeter) in the dialog, the “p” and “u” values in the formula

\[
p \text{ pixels} = u \text{ units}
\]

will be updated in the dialog. You should then set the name of the units and the real size “u” in this formula.
7.6.6.3 Defining Units by a Known Pixel Size

Alternatively, if you know that a single pixel corresponds to x microns (for example, by calculations based on optical magnification), then you can enter x in the formula

\[ x \text{ units} = 1 \text{ pixel.} \]

You can also obtain the value of x from IPLab variables.

7.6.7 Set Measurements

Set Measurements lets you specify which measurements and statistics will be computed by the commands Measure Segments and Measure ROI. It only sets up internal parameters; it does not affect any image windows.

Besides selecting which measurements will be done, you can also use this command to specify limiting criteria for each parameter (that is, the minimum and maximum values the parameters may have). When measuring segments, any segment that does not satisfy the limiting criteria will not be counted or measured. This allows you to look for particles with specific shape, size, or position parameters.

![Set Measurements Dialog Box](image)

Set Measurements Dialog Box

To set limits on a parameter, check the Limit checkbox for that parameter. To ignore limits on a parameter, clear the Limit checkbox for that parameter. The values you enter for the upper and lower limits are interpreted in the units which you have assigned to the image with the Measure Segments command. So, if the image is calibrated in microns, the area measurement will report results in square microns, and the limits you enter should be in square microns, as well.

As noted at the bottom of the dialog box, you can type “-INF” for the lower limit or “INF” for the upper limit. Do this if you want to use an upper limit without a lower limit, or a lower limit without an upper limit. For example, the dialog box pictured above limits the mean to between ten and infinity, which is to say, anything above ten.
When you use one of the measuring commands, the measurements will be placed in a table behind all other windows. When you measure more than one parameter, the results are reported in a table with the parameter names in columns across the top of the table, as shown below. Note that the measurement and image numbers start at one, while the row numbers start at zero.

![Measurement Results](image)

Multi-Parameter Measurement Results

Optional Single-Parameter Measurement Results

When you are measuring a single parameter, you can choose a different form of presentation for the results table by checking the Single Measure: Column Ordered box. With this box checked, a single row in the measurement results is generated each time you perform Measure Segments. The table shows the name of the single measured parameter with the segment number at the top of each column. One application of this form of results reporting is in time series analysis. In each column, you keep track of one parameter value on a single segment at different times. With multiple columns, you can keep track of the parameter on multiple segments. These results are then in a form that can be plotted directly as parameter value versus image number with QGraph.

When using Single Measure, the Max. Number of Segments value in the dialog tells IPLab how wide to make the measurement results window. When set, this limits how many segments each application of Measure Segments will measure.

If you check the box labeled Classify According To, a new Floating Point image will be created in addition to the measurement results window. In this “*.Classes” image, the pixels on each particle are given a single value. The value for each particle is the value of the chosen measurement parameter for that particle. You can select which one of the measured parameters to use via the pop up list below the checkbox. The image is also color coded with a color table that helps you distinguish between the various parameters.

### 7.6.7.1 Densitometry

**Sum, Mean, Min, Max, RMS, Stand. Dev.**

The formulas used to compute the density, or intensity, measurements are as follows. Let \( N \) = Number of pixels in the region, and \( I_{xy} \) = pixel values within the region. Then:

\[
\text{Sum} = \sum I_{xy} \quad \text{Mean} = \frac{\sum I_{xy}}{N} \quad \text{RMS} = \left[ \frac{\sum I_{xy}^2}{N} \right]^{1/2} \quad \text{Standard Deviation} = \left[ \frac{\sum I_{xy}^2}{N} - \text{Mean}^2 \right]^{1/2}
\]

### 7.6.7.2 Shape

**Area**

IPLab reports the area using the units specified by the Set XY Units command (described on page 235). The Area calculation treats a particle as an aggregate of square pixels.
Perimeter

*IPLab* measures perimeters using the units specified by the Set XY Units command (described on page 235). Set Measurements offers two methods for calculating perimeters:

- The **Pixel Method** measures the perimeter of an object by counting each side of each pixel that makes up the perimeter of the object. This calculation treats a particle as an aggregate of square pixels. Therefore, a segment consisting of a single pixel has an area of one and a perimeter of four. This definition of perimeter gives the most consistent results across all scales of magnification; however, it may give results you do not expect for a particle with many diagonal sides.

- The **Line Method** measures the diagonals of an object more carefully than the pixel method. It measures the perimeter of an object by 1) measuring across the diagonal of pixels that make up non-horizontal and non-vertical sides, and 2) using the pixel method to measure sides that are parallel with the image’s edge.

The following example shows the measurements for a 20 x 20 pixel square that has been rotated 45° (using **Rotate & Scale: Nearest Neighbor**):

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Area:</td>
<td>400</td>
</tr>
<tr>
<td>Pixel Perimeter:</td>
<td>80</td>
</tr>
<tr>
<td>Line Perimeter:</td>
<td>80</td>
</tr>
</tbody>
</table>

**Radial S.D.**

The Radial Standard Deviation is the standard deviation of the distance from the centroid to the outer edge of the ROI or segment, reported as a percentage of the mean radius. It is a measure of how much like a smooth circle the ROI or segment is. Below are two particles that can be distinguished by this parameter. The Radial S.D. for the particle on the right, which is nearly a circle, is close to zero. The particle on the left is also nearly a circle, but its edge is substantially rougher, so its Radial S.D. is somewhat larger. In general, a Radial S.D. value of less than 2 - 3% can be considered a smooth circle.

Particles that can be differentiated by Radial S.D.

**2 Mom.-x, 2 Mom.-y, 2 Mom.-xy**

These are the second order moments of the shape. They are computed as follows:

\[M_x = \frac{\sum (x-c_x)^2 S_{xy}}{\text{area}}\]

\[M_y = \frac{\sum (y-c_y)^2 S_{xy}}{\text{area}}\]

\[M_{xy} = \frac{\sum (x-c_x)(y-c_y) S_{xy}}{\text{area}}\]

where \(S_{xy}\) has value 1 inside the particle (ROI or segment), and value 0 outside, and \((c_x, c_y)\) is the position of the particle (the shape centroid).
Major Axis, Minor Axis, Angle, Eccentr.

These parameters are derived from an ellipse that is fitted to the particle (ROI or segment). The major axis result is the length of the longest cord that goes through the center of the ellipse, and the minor axis is the length of the chord that is perpendicular to the major axis. You can use the minor axis as a measure of the minimum diameter of a particle. It is actually a measure if the minimum diameter of the fitted ellipse, which may differ slightly in size from the particle itself. We used the minor axis as a criterion for accepting or rejecting small particles in one of the tutorials.

The angle parameter is the angle between the major axis and the x-axis (horizontal) for the fitted ellipse. It has values between 0 and 90°. The eccentricity is a measure of the circularity of the ellipse. It is 0 for perfect circles and approaches 1 for extreme ellipses.

The eccentricity for a given shape may be near 0, while the Radial S.D. may be very large. Such a situation tells you that the shape is basically a circle with a rough edge, like the shape shown in the picture above next to the circle. However, if both eccentricity and Radial S.D. are large, then the shape is far from a circle.

The formula for eccentricity is

\[ e = \frac{\sqrt{a^2 - b^2}}{a} \]

where \( a \) is the major axis and \( b \) is the minor axis of the ellipse.

7.6.7.3 Position

Centroid, Min and Max Location

The position parameters Centroid, Min. and Max. Loc. report distance from the upper left corner of the image, in the units specified by the Calibrate Distances command.

7.6.7.4 Intensity Moments

Weighted Centroid and 2nd Moments

These are similar to the centroid and shape moments, except that the actual pixel intensities on the particle are used in forming the sums. The formulas are:

\[ a_x = \frac{\sum x I_{xy}}{S} \quad \text{(x-Centroid)} \]
\[ a_y = \frac{\sum y I_{xy}}{S} \quad \text{(y-Centroid)} \]
\[ M_x = \frac{\sum (x-a_x)^2 I_{xy}}{S} \]
\[ M_y = \frac{\sum (y-a_y)^2 I_{xy}}{S} \]
\[ M_{xy} = \frac{\sum (x-a_x)(y-a_y) I_{xy}}{S} \]

where \( I_{xy} \) is the pixel value at location \( x,y \), and \( S \) is the sum of the pixel values: \( \sum I_{xy} \).
7.6.8 Measurement Options

This command lets you set certain options to be used when performing Quantify Segments, Measure Segments and Measure ROI. This command only sets up internal parameters; it does not affect any images.

Measurement Options Dialog Box

First, you must specify which color of segments to measure, whether to erase segments that fail the limits you may have set in the Set Measurements dialog, and whether to ignore segments that touch the edges of the ROI.

You can specify how to annotate each segment after it is found. Only segments passing the limit criteria defined in Set Measurements will be annotated. You can have outlines and/or numbers (labels) drawn on the segments. The annotation is done in the drawing layer of the overlay, so the outlines and numbers can be deleted later. Some extremely large segments will be outlined by the rectangle that encloses the segment, rather than by the actual outline.

The Measurements Results window can have a label attached to each row. The label can include either the name of the image that generated the measurement on that row, or the date and time that the measurement was performed, or both.

You also specify the name of the window in which to place the results or Measure ROI or Measure Segments and the maximum number of measurement results that window can hold. When Measure ROI or Measure Segments is run, they look for a window with this name, having the width and height which match the number of measurements you wish to perform. If such a window is not found, those commands create a window with the name you have specified in this dialog.

7.6.9 Measure ROI

This command applies the measurements specified by Set Measurements to the region of interest. After performing the measurements, the results are stored in a results window with the name specified in Measurement Options. If the results window does not already exist, it is created behind all other windows. If the results window does already exist, then the measurements are appended to the list of existing measurements. When you are finished measuring, use the Window menu to bring the results window to the front.

You may freely mix commands to measure segments and the ROI. All measurement results are placed in the same results window until you use Measurement Options to change the name of the results window.
7.6.10 Measure Segments

This command measures the non-touching segments in the active image window. Segments can be created as a result of the Segmentation command or manually using one of the segment tools on the Tools palette. Only those segments which pass the limit criteria specified in the Set Measurements and Measurement Options commands are measured. You may define the ROI to be some portion of the full image and measure only the segments that fall within that region. Segments that do not match the criteria are erased from the overlay if you specified that option in Measurement Options.

You specify which measurements to perform and what the name of the results window should be using Measurement Options.

After the segments are measured, the results are stored in a results window with the name specified in Measurement Options. If the results window does not already exist, it is created behind all other windows. If the results window does already exist, then the measurements are appended to the list. When you are finished measuring, use the Window menu to bring the results window to the front.

If you have specified more than one parameter to measure in the Set Measurements dialog, or you do not have the Column Ordered box checked in that dialog, then the results table contains each measurement value in a different column. The first column of the results window is the measurement number. This number matches the number that is optionally drawn on the overlay for each segment that meets the segment criteria.

If you measure only a single parameter while using Set Measurements' Column Ordered option, then the results table contains each segment in a different column, with the column label indicating the type of measurement and the segment number. In this case, the first column of the results window is just a running count of the images which have been measured.

7.6.11 Quantify Segments

This command performs a few global measurements on the non-touching segments in the active image window. Only those segments which pass the limit criteria specified in Set Measurements and Measurement Options command are analyzed. You may define the ROI to be some portion of the full image and apply this command to those segments that fall within the ROI. Segments that do not match the criteria are erased from the overlay if you specified that option in Measurement Options.

This command has no dialog. It performs a function similar to Measure Segments, except that it performs measurements on all of the segments as a whole:

- Count of all of the segments that satisfy the limit criteria
- Area information: Total and average area of all of the segments that pass the criteria and the area as a percentage of the total ROI area
- Intensity statistics: Min, max., sum (integrated density) mean, RMS, and standard deviation

These values are placed in a window called Segment Totals. If an appropriate window with that name does not already exist, it is created first. If that window already exists, the measurements are appended to the list.
When you are finished using this command on as many different windows as you require, use the Window menu to bring the Segment Totals window to the front.

Segments can be created using the Segmentation command or by drawing into the overlay with one of the segment tools on the Tools palette.

### 7.6.12 Histogram

The Histogram command gives you several options for computing a histogram of the data within the active data window.

![Histogram Dialog Box](image)

You must specify the value of the **Minimum Bin** and **Maximum Bin**. Pixels with values outside the values of the **Minimum** and **Maximum Bin** will be ignored. If you click **Use Data Minimum** or **Maximum**, then the corresponding value will be taken from the entire image, or the ROI, depending on which you choose at the bottom of the dialog. Alternatively, you may enter any floating point value (or any IPLab variable) you wish for the values of the **Minimum** and **Maximum Bins**.

You must specify the **Number of Bins** to use. The default is 256 bins, which is the best number for Byte data. The maximum is 32,760 bins. If you pick \((\text{Max} - \text{Min}) + 1\), the program will compute the number of bins for you according to this formula:

\[
\text{No. Bins} = \text{Max. Bin} - \text{Min. Bin} + 1
\]

You may compute the histogram on the entire image (on Entire) or on just the data within the ROI (on ROI).

When you press **OK**, the histogram will be displayed in a floating point window with the name `name.Hist` where `name` is the name of the active window. See the tutorial on Real Time Image Analysis for more information on this. The histogram data will be displayed as a QGraph plot. Viewed as text, it contains two columns. The first column shows the histogram bin value, and the second shows the number of image pixels with that value (i.e., in that histogram bin).
7.6.13 Row/Col Plot

This command generates a 1D data window that is a sum or average along selected rows or columns of the image. You may elect to obtain the result on the entire image (independent of the ROI) or on just the data within the ROI.

The results are placed in a window with name name.Sum or name.Average where name is the name of the active image window. If the output window already exists, then the contents of the existing window will be overwritten. This feature is useful for real time analysis while you are capturing images. See the tutorial on Real Time Image Analysis for more information on this.

Row/Col Plot Dialog Box

7.6.14 Contours

The Contours command segments boundaries between different pixel intensities in the front image.

Contours Dialog Box

When contouring color images, the first thing you should do is to pick from the Channel pop-up box which color channel you want to segment. You can pick from the RGB, HSV, YIQ, and R\(_G\)B\(_s\) systems of color coordinates, which are described starting on page 130. Non-color images have only the Gray channel.

Use the long, narrow histogram at the top of the Contours dialog to place the contours. The numbers to the left and right of the histogram limit the channel values displayed in the histogram. You can zoom in on part of the histogram by using the Absolute Range Based On control or by typing new values.

The Absolute Range Based On pop-up box sets the lower and upper limits of the histogram for you. You can use the minimum and maximum values found in the current Frame, the whole Sequence, in the Saturated Frame, or in the Saturated Sequence. The Saturated options exclude the bottom and top 1% of values from the range.

You can also type new minimum and maximum values into the text boxes on either side of the histogram. After typing a histogram boundary, hit the Tab key to update the histogram. The absolute range is based on Manual when you type numbers into the text boxes.
To place some contour lines, choose either the ranged contour markers, [(marker)], or the single contour marker, [marker]. Drag the marker to the histogram and drop it within the box over the desired value. The marker(s) will then appear below the histogram, and symbolic contour lines will appear on the histogram. Drag the markers where you want them, or type intensity values into the markers’ text boxes. After typing a marker position, click **Apply** or hit the Tab key to make the marker move. When you select one marker, the others will be grayed out.

After positioning markers, click **Apply** to contour the image. Click **Clear All** to remove markers from the histogram. Click the trashcan to delete the selected marker, or drag the marker to the trashcan.

The ranged contour markers let you place multiple contours at regular intervals. Choose from the **Range Settings** section: The **Lines** option controls the number of contour lines, which includes the two lines above the two range markers. The **Step** option places as many contours as possible at regular intervals between the two markers, beginning with a contour at the lower range marker.

Pick the color of the contour segments using the **Segment Color** pop-up box. When you click **OK**, this color will be used for all contours in the histogram. If you want contours with different colors, use the **Contours** command multiple times, with a different color each time.

When you click **OK**, all contours in the histogram will be painted on the image’s segment layer. If you click **Cancel**, then no new contour segments will be added to the image.

The **Contours** command does not remove existing contours, though new contours will replace old ones when they overlap. The same **Analyze** menu commands and **Tools** palette items that affect segments can remove and edit contours.

Since contours are segments, they can be measured with the **Quantify Segments** and **Measure Segments** commands.

### 7.6.15 Transforms

The **Transforms** command has a dialog which allows you to select Fourier or Cosine transforms and their inverses. The dialog also prompts you for the names of two windows if you are doing a Fourier transform. If you want to do a forward Fourier transform on a single window (*i.e.* the **Inverse** box is not checked), you can enter a single window name. If you leave both window names blank on the forward transform, then **IPLab** uses the active image window as the real part in a complex forward Fourier transform. The forward transform may be applied to any of the data types except Color16 or Color 24.

Fourier transformation can be performed to yield the result in either Magnitude/Phase or Real/Imaginary formats. For Fourier transformation, the results are placed in two Floating Point data windows containing either Magnitude and Phase or Real and Imaginary FFT output. The window names are derived from the original window name by appending `.FFTMag` and `.FFTPhase` or `.FFTRe` and `.FFTIm`, respectively. The Cosine transform yields a single Floating Point window. Its name has `.Cos` appended .to the name of the active window.

In either case, if the output windows already exist, then the contents of the existing windows are overwritten. This feature is useful for real time analysis while you are capturing images. Please see Tutorial 9: Real Time Image Analysis, on page 117, for more information.
Transform Dialog Box

The Fourier and Cosine transformations require that the ROI size be a power of two in both directions; e.g., 128 x 512, 1024 x 1024, etc. Therefore, the transforms use a size (in each direction separately) which is the nearest power of two greater than or equal to that of the ROI. If the transform size exceeds that of the ROI, the region beyond the ROI is zero filled before performing the FFT.

When performing an inverse FFT operation, the dialog requires you to name two windows which contain Floating Point data, and again the ROI dimensions must be powers of two.

Important: The FFT results, both Magnitude/Phase and Real/Imaginary, appear with the DC term centered at (width/2, height/2) and with the $f_y$ component increasing downward. This differs from the standard image coordinate system referred to in the manual, but only in the location of the (0,0) term. Since the Status palette reports the x,y coordinates of the cursor location with (0,0) in the upper left hand corner, you cannot immediately read the $f_x$-$f_y$ component indexes of an FFT from the Status palette. Furthermore, you should note that, with DC at (width/2, height/2), the highest absolute frequencies occur along the top and left edges of the FFT images windows. Points along the top and left edges are width/2 and height/2 samples from DC whereas points along the right and bottom edges of the window are only width/2 - 1 and height/2 - 1 samples from DC. If you wish to use the Status palette to determine the frequency index of a specific location in the FFT plane, use the Barrel Shift operation (Math menu) to shift the DC term up to the (0,0) location in the FFT plane. This gives a Fourier plane representation that has $f_x$ increasing to the right, $f_y$ increasing downward and DC in the upper-left-hand corner.

7.6.15.1 Scripting the Transforms Command

When scripting, you may enter any names for the inverse FFT operands, even if these windows do not exist at the time of editing the script. This allows the script to use windows that may be created earlier in the running of the script.
7.6.16 **Calibrate Intensities**

When you grab an image, pixel intensities depend on sensor and lighting characteristics. These characteristics may be non-linear. You can use the **Calibrate Intensities** command to create a new image with pixel intensities that match a set of standard intensities in a known target. First, grab the image (or open an existing image). It is best to choose an image with a pattern of known intensity standards, like the one shown below, but the standards may be on any image.

![Image with Standard Intensity Pattern on It](image)

1. First, measure the intensity values within the standard target areas to see how the imaging system has changed their values.
   
   a. Use the **Set Measurements** command to measure only the **Mean** value.
   
   b. Next, select an ROI within the known standards area and choose **Measure ROI**. *IPLab* will create a new window with the ROI's mean pixel intensity in its second column. (The **Measurements Results** window is created behind all other windows.)
   
   c. Drag the ROI to the next area of known intensity by pressing the Option key and click-dragging the ROI.
   
   d. Repeat the ROI measurement.
   
   e. Continue until all of the known intensities from the standard have been measured.

2. Next, bring the **Measurements Results** window to the front. Notice that the mean values of all of the regions you selected are stored in the second column. You can enter the true values (standards) for each of these measured regions in the first column, using the **Set ROI Value** command.

   a. Click on a cell in the first column of the **Measurements Results** table so it is highlighted and underlined in red.
   
   b. Choose the **Set ROI Value** command from the **Edit** menu and enter the standard value that corresponds to this measurement.
   
   c. If you like, you can use the **Set Row/Col Label** command (**Edit** menu) to make the first columns heading more appropriate.

When the standards have all been entered, you will have a function table that tells how each intensity from your standard target is mapped by the imaging system into the computer image. You may want to save this worksheet for later reference.
The actual calibration is performed by the **Calibrate Intensities** command. The dialog asks for the name of the window that contains the image to be calibrated and the names of the windows that contain mean measurements and standard values. If you use the procedure described above, both the Measurements and the Standards values will come from the same window (columns 1 and 0, respectively), but this is not required. In fact, you may want to store the standard values in a separate file since they don’t usually change (assuming you always use the same standard target). You must also select the type of curve with which you want to fit to the data.

![Calibrate Intensities Dialog Box](image)

When the curve’s fit looks appropriate, click **OK**. The new image’s Floating Point data is the result of mapping the original image data into calibrated values by using this function. You can now perform intensity measurements on the calibrated image with floating point accuracy.

Since this command performs an interpolation, it is possible to generate negative image values, even when all values should be positive. To guarantee that output image values are non-negative, check the box labeled **Set Negative to Zero**.

Although floating point images are highly precise, they take up a lot of memory. You may want to change the type to another data type that occupies less memory, but only if you can do so without corrupting the calibration you just performed. If the standard values are in the range 0-255, it is likely that the new Floating Point image will have values in this range as well. Check the **Max** and **Min** values displayed in the **Status** palette to make sure this is the case. If so, you may use the **Change Data Type** command (Math menu) to change the image to Byte type. Use the **Clip** option in **Change Data Type** so the data values are merely truncated to integer, and not mapped by the scaling function.

Remember, however, that there are only 256 possible data values in Byte type, while there are millions of data values in Floating Point type. Subsequent intensity measurements on the Byte data will be less precise than similar measurements on the Floating Point data. Alternative data types are the Integer and Unsigned16 data types, if the calibrated image has values in the range 0-32,767 or 0-65,535, respectively.

### 7.6.17 Dispose Segment Layer

This command erases all of information in the active window’s segment overlay. It also clears the memory allocated to the overlay.
7.6.18 Extract

This is a hierarchical menu that gives several options for extracting information from the active data window.

**Extract Color Table:** This creates a 3 x 256 data window of Byte type containing the red, green and blue values for the window’s color table. Note that for a monochrome image, red = green = blue. The new window will have .CLUT appended to the active image window’s name.

...**Rect ROI to 1D:** This command stores the ROI’s data as a single column in a new, 1D window.

...**Linear ROI to 1D:** This extracts the data values along a multi-line ROI through an image, and places the results in a new, 1D window. The new window will have .Slice appended to the active window’s name. You may create the multi-line ROI with the Define ROI command under the Edit menu or with the mouse using the multi-line ROI tool.

![Image with Linear ROI](image1.png) ![Example Output from Extract: Linear ROI to 1D](image2.png)

...**ROI Boundary:** This command creates an image window that has 3 columns. The first two columns contain the x,y coordinates of the pixels along the ROI boundary. The third column contains the data values at each of the points on the boundary. The suffix .Bnd is appended to the image’s name.

![Image with Rectangular ROI](image3.png) ![Example Output from Extract: ROI Boundary](image4.png)
**Extract Image Sizes**:  

*Extract Image Sizes* extracts information about the image and the image window. It places this information in a new text window or in specific variables. The dialog box tells you exactly where *Extract Image Sizes* will store each piece of information:

1. Coordinates of the left, top, right and bottom sides of the ROI,
2. Width and height of the ROI,
3. Image width and height,
4. Index of the currently displayed frame (the Z depth),
5. Number of frames in the image sequence, and
6. Coded value for the data type:
   - 0=Byte
   - 1=Integer
   - 2=Long Integer
   - 3=Floating point
   - 4=Color16
   - 5=Color24
   - 6=Unsigned16
7. The XY unit factor, as set by the *Set XY Units* command. This is the number of units per pixel. You can only get this value by choosing the *Variables* radio button.

When you choose *New Window*, this command places all of this information into a text window. It appends the suffix *Sizes* to the original image’s name.

*Extract Image Sizes* is very useful for providing scripts with information about an image. If, for example, your script needs to know the current image’s data type, extract the image sizes to variables. Then use an *If* statement to check the value of variable #212.
7.7  Math Menu

The Math menu provides advanced commands that alter the values in the active data window. In general, the commands in the Math menu can be applied to both 2D and 1D data in any of the data types, including the Variables window, and they operate only on the active window. Here are the exceptions to these rules:

**Swap 0 and 255:** Applies only to Byte type data.

**Complex:** Applies only to Floating Point type data and requires that you name the two windows to which it will be applied.

**Encode-Decomde:** Applies only to Byte type data.

**Split Color Channels,**
**Affine Color Transform,**
**Convert Color Chnls:** Apply only to Direct Color data types.

### 7.7.1 Convert to 8/24 Bit

This command changes the active window to a data type (8 bit or 24 bit) that popular art programs can open, while retaining the appearance of the data. The data values are changed according to the Normalization command’s parameter settings for the active window.

Grayscale data is changed to the eight-bit, Byte data type. Color data is changed to the 24 bit, Color 24 data type.

Because this command does change the data values, use it after your data analysis is complete. The converted data is placed in the original window, so saving your data before using this command is a good idea.

### 7.7.2 Unsigned<>Signed

The Short Integer and Unsigned16 data types both use 16 bits to represent data values. The range of values for Short Integer is [-32768, 32767] and for Unsigned16 is [0,65535]. You will generally use Short Integer when you need 16 bits of dynamic range; however, some camera systems are capable of directly creating Unsigned16 data. When IPLab imports image data, it assumes that any 16-bit data is Short Integer. If you import an image that you know is really Unsigned16, use this command to switch the data type to Unsigned16. The data values are not changed (*i.e.* no bits of the data are changed); only the way IPLab treats the data is changed.
7.7.3 Change Data Type

The Change Data Type command allows you to select a new data type for the active window. This command does not work with Color 24 and Color 48 images. As explained below, this command may alter your data, so save a copy first.

Change Data Type Dialog Box

When you select a new data type that uses fewer bytes than the current data type, some of the window’s data values may be out of the range of the new data type. In that case, the When changing to Lower Type section appears, allowing you to control the conversion.

You may select Clip or Scale and enter the range you wish to use. Clipping sets the value of any data element that is less than the Min parameter equal to the Min, and sets the value of any data element that is greater than the Max parameter equal to the Max. All other data element values are unaffected, except that when converting from Floating to any other data type, the data values are truncated.

Scaling changes the data values of all elements by the following transformation:

\[
\text{new value} = \text{Max} \times \frac{\text{data value} - \text{min data value}}{\text{max data value} - \text{min data value}} + \text{Min}
\]

where the values min data value and max data value are taken from the entire image. Scaling is similar to linear normalization, but the Change Data Type command actually changes the values of the data, while Normalization affects only the display of the data.

7.7.4 Change Color Type

This command allows you to change the data type of Byte, Color 24, and Color 48 images in the active window. The old data is replaced by the new image, so save a copy in advance. When converting between one of the Direct types (Color 24 or Color 48) and Byte, an optimal CLUT is calculated to capture as many of the colors as possible from the original image.
There are several applications for this command. For example, if your color input device only gives you 8-bit data, you may want to convert that 8-bit data into 24- or 48-bit data to perform color analysis in IPLab. Also, by converting your 8-bit images into 24-bit images, you can paste together images which originally had different color tables. This is impossible to do properly if you work with 8-bit images only, since 8-bit images with different color tables are incompatible for pasting. Alternately, you may want to convert your Color 48 images to Color 24 or Byte images as a means of data compression.

### 7.7.5 Subsample

Use this command to decrease the size of a data window by removing pixels or image frames from the original data. You can select a different decimation factor in each dimension, x, y, and z. The dialog also allows you to select a starting point and starting frame for this operation.

For each 2D image frame, the **Start Pixel** denotes the location in the original image of the first pixel to be kept after decimation. In the example figure above, a 3:1 decimation was selected with the **Start Pixel** at coordinates (2,1). If “x” denotes pixel values in the original image and “_” denotes pixels removed from the image, then the array

\[
\begin{array}{ccc}
  x & x & x \\
  x & x & x \\
  x & x & x \\
\end{array}
\]

represents the decimation operation for this example.

The choice of 2:1 subsampling in the z direction with a **Start Pixel** of 0 indicates that you want to keep every other frame from the sequence, starting with the first frame.

This command replaces the old window with a new one of the same data, so take care not to lose any unsaved data.
7.7.6 Interleave Zeros

Use this command to increase the size of a data window by interleaving zero values with the original data. Normally, you follow this command with a Linear Filter command to smooth over the data. By selecting different filters as the second step, you can perform a variety of interpolation algorithms.

The Interleave Zeros dialog allows you to select a starting point for this interleaving operation.

![Interleave Zeros Dialog Box]

The Start Pixel denotes the pixel location in the interleaved image that will contain the first, or upper-left, pixel from the original image. In the example figure, a 2:1 interleaving was selected with the Start Pixel at coordinates (1,0). If “x” denotes the value of the (0,0) pixel from the original image, then the array

\[
\begin{bmatrix}
0 & x \\
0 & 0
\end{bmatrix}
\]

represents the top-left pixels of the interleaved image. With 2:1 interleaving, only the pixel locations in the lightly shaded portion of the mask can be selected. For 3:1 interleaving, any of the nine positions can be used as the Start Pixel. For 1D data, only the top row of the Start Pixel mask has any significance.

This command replaces the old window with a new one of the same data, so take care not to lose any unsaved data.

7.7.7 Sort Data

Use this command to sort tables of data (that is, images viewed as text). You specify which column to sort, and the rows of the table are re-arranged so that column is in ascending or descending order.

![Sort Data Dialog Box]
7.7.8  Barrel Shift

Using the **Barrel Shift** command shifts all of the data in the active window so that the upper left corner of the ROI is moved to the (0,0) location in the window. Since the location of the ROI is not affected by this command, you may shift the data in equal steps by repeated applications of this command without having to change the ROI.

Before being barrel shifted.
Note the selected bay and river

After being barrel shifted.
Bay and river are now in upper-left corner

In the example, the top-left corner of the ROI (left window) is located in the upper left of the picture portion of the window. After barrel shifting, the new image (right window) has this top-left point moved to (0,0) and the rest of the image is shifted accordingly. Only the location of the top-left corner of the ROI is important. The other edges of the ROI rectangle are ignored for this command.

7.7.9  Swap 0 and 255

By Macintosh convention, most programs display Byte type images so that pixels with data value 0 appear white (*i.e.* RGB = (255,255,255)) and pixels with data value 255 appear black (*i.e.* RGB = (0,0,0)). So, in the color look up tables used by most image processing programs, 0 corresponds to white and 255 corresponds to black. However, that convention is opposite to what is normally used in the scientific community for monochrome image display and processing. Therefore, **IPLab** uses the scientific convention and does not adhere to this Macintosh convention for display.

Nevertheless, you may import image data into **IPLab** which does adhere to the 0=white, 255=black convention. That includes images captured by other programs that do not use scientific conventions. This **Swap 0 and 255** command provides a quick way to change the values of all pixels having the values of 0 or 255. This command operates only on Byte type data. Each data element with a value of 0 is set to the value 255 and each data element with a value of 255 is set to the value 0. All other pixel values will be unaffected.
7.7.10  Binarize

Use the Binarize command to create an image of segment locations (particularly for isolating segmented data).

![Binarize Dialog Box]

Binarize Dialog Box

Choose a segment color by clicking on the For Segment Color box. When you click OK, the data values under those segments will be changed to one. All other data values will be changed to zero. The resulting image will replace the original image unless you check the Create New Window option. The resulting image will have .bin appended to its name.

7.7.10.1  Using Binarize

This is a good way to isolate segmented data, e.g. for mathematical operations. As a simplified example, you can use Binarize to remove the background, leaving only the data. First, segment the data and use the Binarize command with the Create New Window option. Then use the Image Arithmetic command to multiply the original image with the new binarized image. All data under the chosen segments will be multiplied by one, and will remain. All other data will be multiplied by zero.

Note that you can set unsegmented data to one, and all segmented data to zero, by choosing Transparent as your segment color.
7.7.11  Pixel Arithmetic

*IPLab* provides a large collection of functions that change the value of each data element in the ROI based only upon the original value of that element.

### Pixel Arithmetic Dialog Box

The **Parameters** \(a\), \(b\), and \(c\) store the values of the constants. Toggling a # button to **Var** lets you enter the index of a variable in the **Variables** window instead of a number.

Several items in the dialog require further explanation.

#### 7.7.11.1 \[ \frac{a \ Log_2 |x|}{c} + b \]

Notice that the logarithm is performed base 2. To get log base \(e\) or log base 10, use the following values for \(a\) and \(c\):

- \(\log_{10} x: \ a = 30103, \ c = 100000\)
- \(\log_e x: \ a = 69315, \ c = 100000\)

#### 7.7.11.2 \(a^{th}\) bit of \(x\) (Bitplanes)

The result of this operation is the \(a^{th}\) bit of the binary representation for \(x\) where:

- \(0 \leq a \leq 7\) for Byte type data,
- \(0 \leq a \leq 15\) for Short Integer and Unsigned16 type data, and
- \(0 \leq a \leq 31\) for Long Integer type data.
7.7.11.3  \((ax + b) \mod c\) (Modulo Arithmetic)

The result of this operation is the remainder of \((ax + b)/c\).

This result has the same sign as \((ax + b)\).

7.7.11.4  Logical NOT(x)

The result of this operation is a bitwise NOT on the data word x.

7.7.12  Image Arithmetic

Image Arithmetic performs point by point mathematics on the ROIs of two input windows. Within the ROI of the two images, the corresponding points will be added, subtracted, multiplied, etc. together. The operations are of the form:

\[
\text{Output} = \text{Constant}_1 \ast \text{Image}_1 \text{ (operator)} \ast \text{Constant}_2 \ast \text{Image}_2
\]

where (operator) is the selected arithmetic process: +, -, *, /, Max, Min, AND, OR, XOR.

The dialog prompts you for two input window names (the operands), both of which must be the same data type. The two images can be different sizes, but their ROIs must be the same size. The ROIs need not have the same relative positions within their respective windows. In fact, you may perform Image Arithmetic between 1D and 2D data windows, as long as the ROIs in the two windows have the same width and height.

Choose the arithmetic operator from the pop-up box between the two window names. The Max and Min options return the maximum and minimum value, respectively, found at each pixel in the two images. AND, OR, and XOR are bitwise operators.

Selecting the / (Divide) option activates the Value of Zero Divide section, where you can specify what value division by zero will give. For any pixel with value 0 in the denominator image, the resultant image value will be replaced with the number entered here.

The boxes to the left of the window names contain the constants that will be multiplied against each ROI’s values. Enter these constants directly or let IPLab obtain them from variables by toggling the # to Var.
The **Place Results In** section lets you either overwrite the data in the first input window or place the results in a new image window.

### 7.7.13 Complex

**IPLab** assumes that complex data is stored in a pair of separate windows of Floating Point type. That is how the **Fourier** options of the **Transforms** command (**Analyze** menu) generate their output. The two windows may contain either the real and imaginary parts or the magnitude and phase components of the complex data.

The dialog for the **Complex** command allows you to select one of three operations on the data in these window pairs. Complex multiplication requires you to enter the names of four windows which hold the real and imaginary parts of two complex images. The multiplication result overwrites the data in the two **A Operand** windows. Neither of these **A Operand** windows need to be the active window to perform this command.

If you choose to convert from a magnitude and phase representation to the real and imaginary representation or vice versa, you need only enter the names of two windows. Note, however, that one of the results of this option is that the source windows are overwritten by the results obtained. For example, in converting from magnitude/phase to real/imaginary, the image containing the magnitude data ends up containing the real part while the image containing the phase information now contains the imaginary part. Similarly, a conversion from real/imaginary to magnitude/phase results in the original real window containing the resultant magnitude and the original imaginary window containing the resultant phase.

You choose operands for the **Complex** command by either typing in a window name or using the pop-up menus provided.

You may use this command on any two floating-point windows. They don’t need to be generated by the **Fourier Transform** selection in the **Transform** command.

![Complex Dialog Box](image)
The formulas for complex multiplication are as follows:

\[\text{Real}(A) = \text{Real}(A) \times \text{Real}(B) - \text{Imaginary}(A) \times \text{Imaginary}(B)\]
\[\text{Imaginary}(A) = \text{Real}(A) \times \text{Imaginary}(B) + \text{Real}(B) \times \text{Imaginary}(A)\]

The conversions between real/imaginary and magnitude/phase use the following formulas:

\[\text{Real} = \text{Magnitude} \times \cos(\text{Phase})\]
\[\text{Imaginary} = \text{Magnitude} \times \sin(\text{Phase})\]
\[\text{Magnitude} = \sqrt{(\text{Real})^2 + (\text{Imaginary})^2}\]
\[\text{Phase} = \arctan(\text{Imaginary}/\text{Real})\]

The phase results from the last conversion are in the range \((-\pi, \pi)\).

Although complex division is not provided as a separate option, it can be performed using a combination of complex multiplication and the \textit{Image Arithmetic} command.

When scripting, you may enter any window name for an operand, even if the named window does not exist at the time of editing the script. This allows the script to use windows that are created earlier in the script.

### 7.7.14 Set Pattern

Four different patterns of values may be added to the data in the ROI of the active window. The pattern is generated in the data type of the active window.

**Patterns:**
- **Horizontal Ramp:** A linear ramp of data with values between \(a/c\) and \(b/c\) is added to each data element in the ROI. If \(a/c < b/c\), the ramp increases from left to right across the window. Otherwise, it decreases from left to right across the window. If the active window contains 2D data, then all of the pixels have the same value along any vertical line in the window.

**Add Pattern** Dialog Box

**Horizontal Ramp:** A linear ramp of data with values between \(a/c\) and \(b/c\) is added to each data element in the ROI. If \(a/c < b/c\), the ramp increases from left to right across the window. Otherwise, it decreases from left to right across the window. If the active window contains 2D data, then all of the pixels have the same value along any vertical line in the window.
**Vertical Ramp:** A linear ramp of data with values between \(a/c\) and \(b/c\) is added to each data element in the ROI. If \(a/c < b/c\), the ramp increases from the top of the window to the bottom. Otherwise, it decreases from top to bottom. If the active window contains 2D data, then all of the pixels have the same value along any horizontal line in the window.

**Uniform Noise:** Uncorrelated uniform random noise is added to each data element in the ROI. Each random value is in the interval \([a/c, b/c]\).

The seed for the random number generator is taken from the system clock at the time the command is executed. This produces a different noise realization each time that the command is executed. In order to use the same noise data for different experiments, you should add a noise pattern to a blank window, save the window, and then add the noise to other windows using the **Image Arithmetic** command. The length of the pseudorandom sequence is approximately \(2^{32}\); that is, \(2^{32}\) uncorrelated numbers are generated before the sequence starts to repeat itself.

**Gaussian Noise:** Uncorrelated Gaussian random noise is added to each data element in the ROI. The mean and standard deviation of the Gaussian noise source are \(a/c\) and \(b/c\), respectively.

As with **Uniform Noise**, the seed for the random number generator is taken from the system clock when the command is executed. This produces different noise realizations each time that the command is executed. In order to use the same noise source for different experiments, you should add a noise pattern to a blank window, save the window, and then add the noise to other windows using the **Image Arithmetic** command. The length of the pseudorandom sequence is approximately \(2^{32}\).

Although the values \(\pm\infty\) are theoretically possible data values for the Gaussian distribution, these values have been excluded from the possible output values.

The **Parameters**: \(a\), \(b\), and \(c\) control the range of values present in the added pattern. Toggle a #/Var button to enter a number or the index of a variable in the **Variables** window.

**Data Type Limits:** Click this button to fill in the \(a\) and \(b\) parameters based on the limits of the front image's data type.

**Add to Existing Data:** When this option is unchecked, the new pattern replaces the current image.

When this option is checked, **Set Pattern** adds the intensity values of the new pattern to the values of the current image.

**Do All Frames:** When this is checked, **Set Pattern** acts upon all frames within an image sequence, instead of only setting a pattern within the current frame.
7.7.15 Encode-Decode

This command may only be applied to Byte type data. **Encode-Decode** is used to apply an arbitrary function to your data. The functional relationship is specified by a 1D data window of Byte type with 256 elements. This array describes how to map values from 0-256 into output values in the same range. The **Encode-Decode** dialog allows you to specify whether you wish to perform encoding or decoding and provide the name of the 1D data window which contains the mapping. **Next Window Down** uses the window beneath the front-most (active) window. **Bottom Window** uses the bottom-most window. These options are most useful in scripts where you may know the relative position of your windows. See the **Change Window** command (Window menu) for a more complete description of these options.

![Encode-Decode Dialog Box](image)

**Encode-Decode** Dialog Box

It may help to understand the operation of this command by considering the selected 1D data as a function \( y = f(x) \). The result for each data element in the ROI is then given by:

\[
\text{result} = \begin{cases} 
  f(\text{data value}) & \text{for Encode} \\
  f^{-1}(\text{data value}) & \text{for Decode.} 
\end{cases}
\]

The inverse function is defined as follows

\[
f^{-1}(y) = \text{smallest value of } y \text{ such that } f(x) \geq y, \text{ if } f(0) \leq f(255), \text{ i.e. non-decreasing functions}
\]

\[
f^{-1}(y) = \text{smallest value of } y \text{ such that } f(x) \leq y, \text{ if } f(0) \geq f(255), \text{ i.e. decreasing functions.}
\]
7.7.16  Split Color Channels

This command splits the active window’s color image data into its three component channels. Split Color Channels will operate on Color 24 and Color 48 images. For more information about color channels, please see the Data Types section on page 129.

Select one of the five coordinate systems to use when splitting the image data (e.g. R-G-B). Choose the color channels you want (for example, Red and Green); you do not have to select all three channels. When you click OK, a separate window will be created for each selected channel. Color 24 images will be split into Byte type images; Color 48 images will be split into images of type Unsigned16. The original, color image will remain open.

The name of the new color channel windows will be formed by appending the corresponding letter designation to the name of the original color window. The window containing the yellow component of CMY uses Ye as the extension to avoid possible confusion with the Y component of YIQ. R-G-B and C-M-Ye channel images will be pseudocolored to represent their original channel.
7.7.17 Merge Color Channels

7.7.17.1 Merge Tab

This command is used to form a new Color 24 or Color 48 image out of its component channels. For more information about the color channels that make up a color image, please see the Data Types section on page 129.

![Merge Tab of the Merge Color Channels Dialog Box](image)

From the **Merge Channels** section, select one of the four coordinate systems (RGB, YIQ, HSF, or CMY) to use when merging the images. Next, pick the input windows that will be treated as red, green, and blue. In the three **Window** boxes, type the window names or pick them from the pop-up lists.

The **Merge Raw Data...** option lets you choose to produce a color image from the original (raw) data or from the normalized image as it is currently displayed. This option works upon Byte data when creating Color 24 images. It works upon Byte and Unsigned 16 data when creating Color 48 images.

- When **Merge Raw Data...** is **unchecked**, the data to be merged will be taken from the normalized (displayed) data. This allows you to produce a good-looking image from images whose displays you have already optimized.

- When **Merge Raw Data...** is **checked**, the command merges the image's raw data and ignores all normalization. This is a good idea if you might want to measure the data later. You can normalize the channels within the color image by using the **Normalization** command (**Enhance** menu).

This chart summarizes how the different data types are merged:

<table>
<thead>
<tr>
<th>Source Type</th>
<th>Color 24 Output</th>
<th>Color 48 Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>Raw or Normalized*</td>
<td>Raw or Normalized*</td>
</tr>
<tr>
<td>Short</td>
<td>Normalized</td>
<td>Normalized</td>
</tr>
<tr>
<td>Unsigned</td>
<td>Normalized</td>
<td>Raw or Normalized*</td>
</tr>
<tr>
<td>Long</td>
<td>Normalized</td>
<td>Normalized</td>
</tr>
<tr>
<td>Float</td>
<td>Normalized</td>
<td>Normalized</td>
</tr>
</tbody>
</table>

* Based on the setting of the checkbox.
Choose the **Final Image** output type (**Color 24** or **48**), and click **OK**. The new color image will be created from the component images, which will remain open.

### 7.7.17.2 Blend Tab

Use the **Blend** tab to:
- Mix a fluorescent signal into a brightfield background image,
- Examine co-localization of fluorescent labels by fusing two grayscale channels,
- Add a grayscale image to an existing color image.

**Blend** Tab of the **Merge Color Channels** Dialog Box

First, choose the source **Fluorescent Image**. Then choose your **Brightfield Image**. The fluorescent source image must be a grayscale image. The brightfield image can be any data type, grayscale or color. Both images can be sequences, but they must both have the same width and height.

Next, type the percentage amount of blending in the **Blend** field. The higher the blending percentage, the fainter the background will be.

Finally, choose the color channel into which the fluorescent source image will be placed. If you choose **Green**, for example, your final image will contain green fluorescent data over the gray background image.

The blended data will always be placed in a new image window.

For co-localization, select a grayscale image of a fluorescent label for each source window. To add a grayscale image to an existing color image (as an additional color channel), select the new image as the **Fluorescent** window, and select the color image as the **Brightfield** window.
7.7.18 Convert Color Channels

This command converts color component images from one color coordinate system to another. For more information about color coordinate systems, please see the Data Types section on page 129.

![Convert Color Channels Dialog Box](image)

**Convert Color Channels** Dialog Box

First, tell *IPLab* how to treat the data in these three windows by selecting the appropriate **From Channels** radio button. *IPLab* will transform the data into the color coordinates specified by the **To Channels** radio button. Second, pick the names of three input windows by clicking on the window list arrows or by typing their names in the text boxes. All three windows must be the same data type.

When you click **OK**, the data currently within the three selected windows will be translated into the new color coordinate system. No new windows will be created; the data will be transformed in-place. However, one-letter designations of the new colors will be appended to the original window names. When converting to CMY, the yellow channel is represented by the extension *Ye* to avoid confusion with the Y component of the YIQ system.

7.7.19 Affine Color Transform

This command allows you to define your own color coordinates as an affine transformation on RGB coordinates.

![Affine Color Transform Dialog Box](image)

**Affine Color Transform** Dialog Box
Choose the **From RGB** radio button to convert red, green, and blue component images to your own color coordinate system. Choose **To RGB** to convert the images back to red, green, and blue.

The top of the dialog prompts you for the components of the transformation matrix and vector. If you select the **From RGB** option, the command will apply the following formula to each pixel in the component input images:

\[
\begin{bmatrix}
  C_1 \\
  C_2 \\
  C_3 \\
\end{bmatrix} = \begin{bmatrix}
  M_{11} & M_{12} & M_{13} \\
  M_{21} & M_{22} & M_{23} \\
  M_{31} & M_{32} & M_{33} \\
\end{bmatrix} \begin{bmatrix}
  R \\
  G \\
  B \\
\end{bmatrix} + \begin{bmatrix}
  D_1 \\
  D_2 \\
  D_3 \\
\end{bmatrix}
\]

If you select the **To RGB** option, the command will apply this formula:

\[
\begin{bmatrix}
  R \\
  G \\
  B \\
\end{bmatrix} = \begin{bmatrix}
  M_{11} & M_{12} & M_{13} \\
  M_{21} & M_{22} & M_{23} \\
  M_{31} & M_{32} & M_{33} \\
\end{bmatrix} \begin{bmatrix}
  C_1 \\
  C_2 \\
  C_3 \\
\end{bmatrix} + \begin{bmatrix}
  D_1 \\
  D_2 \\
  D_3 \\
\end{bmatrix}
\]

In the above formulae, \( M \) is a 3x3 matrix and D1, D2, and D3 are additive constants. Enter values for \( M \) and the constants in the boxes at the top of the dialog.

In the **From Windows** section, enter the names of three input windows by clicking on the window list arrows or by typing their names in the text boxes. All three windows must be the same data type (i.e. Byte, Short Int, etc.). When transforming **From RGB**, these will be treated as the red, green, and blue components. When transforming **To RGB**, these will be treated as the C1, C2, and C3 components.

When you click **OK**, the data will be transformed within the existing windows, so save your data first! To indicate the new color coordinates, one-letter color codes will be appended to the original window names. R, G, and B windows will have C1, C2, and C3 appended to the name; C1, C2, and C3 windows will have R, G, and B appended, respectively.

Often, you will want to apply a transformation to some RGB windows, perform some operations, and then transform back to RGB coordinates. **IPLab** helps you determine the inverse transformation needed to return to RGB. If you click on the **Trial Inverse** button, **IPLab** will attempt to compute the inverse of the transformation appearing in the \( M \) matrix and D vector. The values in the matrix and vector will be replaced with the trial inverse. Obviously, if \( M \) cannot be inverted, or if it is ill conditioned, then the trial inverse may be far from the true inverse.
7.8 3D

7.8.1 General Description and Applications

The 3D commands display and measure three dimensional images. You can acquire 3D image data within the IPLab environment or by importing it from an external device such as a confocal microscope. Applications of the 3D commands include microscopy (both transmitted and multi-probe fluorescence), 3D reconstruction of a sliced object, and visualization of computer generated 3D sets of data.

IPLab simplifies image processing and archiving by acquiring and saving 3D images as different frames of a single sequence (in IPLab and TIFF file format). Once you have acquired an image sequence, you may want to preprocess the data to correct for some common problems. The 3D commands provide you with the following options for processing your data prior to visualization:

- Correct for intensity changes due to photobleaching or light source fluctuations across your data set. This is also needed when combining individual images of slices of varying thicknesses, and therefore different optical densities, into a 3D data set. Examples include tissue slices and TEM (transmitted electron microscope) images.

- Sharpen the data set to enhance the visual detail using an interactive filtering technique.

- Merge up to three grayscale image sequences into one color image sequence in order to visualize spatial coincidence of features that appear at different wavelengths. The grayscale image sequences may be acquired using different spectral filters or may be the result of computer generated data applied to a volume. A common problem when collecting multi-spectral data is that there may be spatial displacements between the images acquired at different wavelengths, therefore the merging command allows you to translate each image sequence to register, or align, the images prior to merging.

Commands to analyze and visualize the data include:

- Generate a mosaic or tiled view of all frames of a data set in a single window for presentation or printing.

- Generate a plot of intensity versus frame number across a region of interest.

- Visualize the volume by looking at internal slices from viewpoints along the X, Y, and Z axes. You can focus your attention on a specific point by clicking directly on the point in 3D space. You can also trace in 3D space using drawing tools to highlight items of interest while observing the image data in several frames. The result is a 3D drawing that can be used for counting or for measurements within the IPLab software.

- Make projections of the data set by rotating the viewing angle around a central axis. The resulting sequence can be animated as a movie or made into a mosaic. The projections work with both grayscale and color data sets. The projection algorithm offers choices that are optimized for fluorescence transmission microscopy (darkfield and brightfield).

- Scripts are provided to illustrate how these commands can be used to create stereo views of your 3D data set. Two examples are provided, one for stereo pairs (side by side), and one for color merging stereo (you will need red/green colored "3D" glasses).
7.8.2 Overview of 3D Visualization

The visualization of 3D volumes obtained with a real world camera and microscope presents a number of challenges. The intensity values obtained form a three dimensional array of voxels. The rendering process tries to convey the three dimensional information into a two dimensional medium, such as your computer screen or a printed page. Each voxel in space contains information such as intensity, opacity and position. In the case of a color object, there are three values for intensity: red, green and blue.

In general, there is too much information available to display in a compact manner. Consequently, in all 3D rendering packages some assumptions about the nature of the data are needed in order to make a 2D presentation. Some packages will resort to displaying surfaces instead of voxel values (appropriate if displaying opaque solids). Others assume that voxels with a similar intensity value belong to a similar object class, with a common transparency value (which may be true when displaying clinical data such as computer aided topography scans or magnetic resonance imaging).

Neither of the above assumptions are appropriate if you are looking through fairly transparent objects (such as cells or tissue). For example, in the case of darkfield microscopy, an object has an intensity related to its light scattering properties (refractive index variations). In the case of fluorescence microscopy, an object with uniform loading has an intensity related to its thickness.

7.8.3 3D Equalize

This command equalizes the intensity level or brightness of all frames in a sequence. It does this in two steps: first, it subtracts a background image or an offset value from all frames in the sequence, and then it applies a linear transformation to every pixel.

The image data in the active window may be of type Byte, Short Integer, Unsigned 16, or Color 24. The operation is always performed in place; that is, it changes the values in the original image. The command operates on the active window and is scriptable.

3D Equalize Dialog Box

Constant: When this option is selected, a constant value is subtracted from each pixel in the sequence. You provide the value in the associated text box. If you do not want to apply background subtraction, enter a value of zero. Each pixel in the sequence undergoes the following transformation:

\[ \text{value}[x, y, n] = \text{value}[x, y, n] - \text{Bgnd} \]

where:

\[ \text{value}[x, y, n] \] is the pixel value at point (x,y) in frame n of the original sequence

\[ \text{Bgnd} \] is the background constant.
Window: When this option is selected, the background image is subtracted from each frame in the sequence. Pick the name of the open background image from the associated pop-up box.

Each pixel in the sequence undergoes the following transformation:

\[ \text{value}_{x,y,n} = \text{value}_{x,y,n} - \text{Bgnd}_{x,y} \]

where:

- \( \text{value}_{x,y,n} \) is the pixel value at point \((X,Y)\) in frame \(N\) of the original sequence
- \( \text{Bgnd}_{x,y} \) is the pixel value at point \((X,Y)\) in the background image.

Equalization is only applied to the sequence if the box labeled **Equalize Sequence Using the ROI Mean** is checked. When equalization is applied, it is always done after background subtraction. This option calculates the average ROI mean of all frames, and then scales the pixel values in each frame so that the ROI mean of each frame equals this value (the average ROI mean). The equalization options are:

**Use Addition Only:** When this option is selected, an additive transformation is used to equalize the sequence.

Each pixel in the sequence undergoes the following transformation:

\[ \text{value}_{x,y,n} = \text{value}_{x,y,n} + \text{ROI}_{Avg} - \text{ROI}_{Local} \]

where:

- \( \text{value}_{x,y,n} \) is the pixel value at point \((X,Y)\) in frame \(n\) of the original sequence
- \( \text{ROI}_{Avg} \) is the mean of pixel values within the ROI of all frames
- \( \text{ROI}_{Local} \) is the mean of pixel values within the ROI of frame \(n\)

**Use Linear Scaling:** When this option is selected, a linear scaling transformation is used to equalize the sequence.

Each pixel in the sequence undergoes the following transformation:

\[ \text{value}_{x,y,n} = \frac{\text{value}_{x,y,n} \times \text{ROI}_{Avg}}{\text{ROI}_{Local}} \]

where:

- \( \text{value}_{x,y,n} \) is the pixel value at point \((X,Y)\) in frame \(n\) of the original sequence
- \( \text{ROI}_{Avg} \) is the mean of pixel values within the ROI of all frames
- \( \text{ROI}_{Local} \) is the mean of pixel values within the ROI of frame \(n\)

**Note:** When Equalizing a Color 24 image sequence, the \( \text{ROI}_{Avg} \) and the \( \text{ROI}_{Local} \) values are determined from the sum of the Red, Green, and Blue color components.
7.8.4  3D Filter

This command applies a tunable sharpening filter to an image or to an entire sequence. While this operation is not exactly equivalent to digital deconvolution, this filter can produce similar results. **3D Filter** sharpens an image using adjacent pixels within the same frame or neighboring frames. The filter mask is a cube with three pixels per side, where the pixel being processed is in the cube's center. For color data types, the filter is applied separately to each of the three color channels.

The **3D Filter** command operates on the active window and is scriptable. The active window must contain Byte, Short Integer, Unsigned 16, or Color 24 data.

---

**3D Filter** Dialog Box

- **Preview Frame:** Pick one of the sequence's frames. This feature lets you preview the filter's effects upon this frame. Changes to **In-Frame Sharpness**, **Adjacent-Frame Haze**, **Gain**, and **Use Adaptive Filtering** are shown in real time within the preview frame. When you change **Low Threshold** or **Preview Frame**, click the **Apply Filter** button to see the effects.

- **In-Frame Sharpness:** This parameter controls the percentage amount of image sharpening provided by adjacent pixels within the current frame.

- **Adjacent-Frame Haze:** This controls how much haze from neighboring frames to remove from the current frame, as a percentage.

- **Gain:** This parameter controls the gain of the filter. **3D Filter** multiplies all pixel values by the gain value after the image is filtered. A gain of one leaves the image unchanged.

In the case of Short Integer and Unsigned 16 data, you may not see the effect of gain because the image is normalized when displayed (see **Normalization** on page 207).
**Low Threshold:**

The threshold is the pixel value at which the filter starts sharpening. Only pixels with values above the **Low Threshold** value are processed.

This is especially useful for processing darkfield images, which have features with pixel values higher than the background. (If your image does not fit this model, you may be able to use the **Invert** command in the Enhance menu before applying the 3D filter.) If you set the **Low Threshold** value higher than the background, then only the features of interest will be enhanced, and not the noise in the background.

**Use Adaptive Filtering:**

Adaptive filtering produces a more natural-looking result for darkfield images by modifying the way the processing is applied. With **Adaptive Filtering**, the level of filtering increases gradually with the intensity contribution of adjacent pixels.

Notice that you can always achieve a linear sharpening filter if you set the **Low Threshold** value to zero and do not check the **Adaptive Filtering** checkbox.

**Before/After:**

Switch back and forth between what the image looked like before and after being filtered. Click **Before** to see the original image; click **After** to see the filtered image.

**Apply Filter:**

Click the **Apply Filter** button to update the filtering of the previewed frame. Click **Apply Filter** when you change **Low Threshold** or **Preview Frame**. Changes to the other options within this box are shown in real-time.

**Do All:**

Choose this to filter all frames in the sequence.

**Do Frames ...:**

Choose this option to only filter this range of frames. The others will be left as they are.

**Release Original Sequence:**

Check this box to automatically close the original image sequence after you filter it.

Note: This option closes the original window without saving any changes!

Click **OK** to filter the image sequence (either the entire sequence or the selected range of frames) for real. If you click **Cancel**, then no filtering will occur.

**Note:**

The processing time required for this command can be substantial. For fastest interactive results on large images, you should select a rectangular ROI before using this command. Smaller ROIs yield quicker results.
7.8.4.1 About Adaptive Filtering

Adaptive filtering produces a more natural-looking result for darkfield images by modifying the way the processing is applied. Without adaptive filtering, the amount of filtering (the Delta value in the formulas below) is a step function. If the intensity contribution of adjacent pixels (the Blur value) exceeds the threshold value, the filter will be applied; otherwise, no filtering is done on that pixel. With Adaptive Filtering, the level of filtering increases gradually with the Blur value. The graphs shown below illustrate the differences between adaptive and non-adaptive filtering (Delta of 0.75, Threshold of 30).

Delta vs. Blur without (left) and with (right) Adaptive Filtering

Given values in the original sequence \( \text{Seq}[x, y, z] \), values in the filtered sequence are determined by the following expression if \( \text{blur}_1 \geq \text{threshold} \):

\[
\text{value}[x, y, z] = \text{gain} \left( \frac{\text{Seq}[x, y, z] - \Delta_1 * \text{blur}_1}{1 - \Delta_1} - \Delta_2 * \text{blur}_2 \right)
\]

where \( \Delta_2 = 0 \) for single frames.

and by the following if \( \text{blur}_1 < \text{threshold} \):

\[
\text{value}[x, y, z] = \text{Seq}[x, y, z]
\]

The variable \( \text{blur}_1 \) is the average value of neighboring pixels in frame \( z \) and is determined by the following expression:

\[
\text{blur}_1 = \frac{1}{8} \left( \sum_{i=-1}^{1} \sum_{j=-1}^{1} \text{Seq}[x+i, y+j, z] - \text{Seq}[x, y, z] \right)
\]

The variable \( \text{blur}_2 \) is the average value of neighboring pixels in adjacent frames. Since the frame of interest may or may not have adjacent frames, four cases must be considered.

1. If the image only has a single frame, then

\[
\text{blur}_2 = 0
\]

2. If the frame of interest (frame \( z \)) is the first frame, then

\[
\text{blur}_2 = \frac{1}{9} \left( \sum_{i=-1}^{1} \sum_{j=-1}^{1} \text{Seq}[x+i, y+j, z+1] \right)
\]
3. If the frame of interest is the last frame, then

\[ \text{blur}_2 = \frac{\left( \sum_{j=1}^{1} \sum_{i=1}^{1} \text{Seq}[x+i, y+j, z-1] \right)}{9} \]

4. If the frame of interest is a middle frame, with frames above and below, then

\[ \text{blur}_2 = \frac{\left( \sum_{j=1}^{1} \sum_{i=1}^{1} \text{Seq}[x+i, y+j, z-1] \right) + \left( \sum_{j=1}^{1} \sum_{i=1}^{1} \text{Seq}[x+i, y+j, z+1] \right)}{18} \]

Expressions for Delta ($\Delta_1$ and $\Delta_2$) change when adaptive filtering is used. When adaptive filtering is not used, $\Delta_1$ and $\Delta_2$ are constants:

\[ \Delta_1 = \frac{\text{sharpness}_{\text{in}}}{100} \]
\[ \Delta_2 = \frac{\text{sharpness}_{\text{out}}}{100} \]

When adaptive filtering is used, $\Delta_1$ and $\Delta_2$ are functions of pixel intensity:

for $\text{blur}_1 - \text{threshold} < k$

\[ \Delta_1 = \frac{\text{sharpness}_{\text{in}}}{100} \left( \frac{\sqrt{\text{blur}_1 - \text{threshold}}}{\sqrt{k}} \right) \]
\[ \Delta_2 = \frac{\text{sharpness}_{\text{out}}}{100} \left( \frac{\sqrt{\text{blur}_1 - \text{threshold}}}{\sqrt{k}} \right) \]

for $\text{blur}_1 - \text{threshold} \geq k$

\[ \Delta_1 = \frac{\text{sharpness}_{\text{in}}}{100} \]
\[ \Delta_2 = \frac{\text{sharpness}_{\text{out}}}{100} \]

where:

$\text{sharpness}_{\text{in}}$ is the In-Frame Sharpness value entered by the user

$\text{sharpness}_{\text{out}}$ is the Adjacent-Frame Haze value entered by the user

The filtering level $\Delta$ is zero at the threshold level, and reaches the maximum value at $k$. The value of $k$ depends on the data type. For Byte and Color 24 data, $k$ is a constant:

\[ k = 127 \] for Byte and Color 24 data

For Short Integer and Unsigned 16 data, $k$ is the midpoint in the range of pixel values in the sequence:

\[ k = \frac{\text{value}_{\text{min}} + \text{value}_{\text{max}}}{2} \]
### 7.8.5 3D ROI Sum Plot

This command measures the sum or mean of intensities within the ROI of a sequence's frames. The command operates on the active window and is scriptable. The active window must be a sequence containing Byte, Short Integer, Unsigned16, Long Integer, or Floating data.

#### 3D ROI Sum Plot Dialog Box

This command provides the following options:

- **All frames:** When this option is selected, the **ROI Sum** or **Mean** is calculated for all frames in a sequence.

- **Custom:** When this option is selected, the **ROI Sum** or **Mean** is only calculated for the frames that you specify. The first frame and the last frame to be included in the calculations must be entered in the associated text boxes.

- **Sum:** When this option is selected, the sum of intensities within each frame's ROI is calculated and displayed in the QGraph.

- **Mean:** When this option is selected, the mean of intensities within each frame's ROI is calculated and displayed in the QGraph.
The measurement results are displayed in a QGraph showing the sum or mean values (on the Y axis) versus the frame number (on the X axis).

An Example QGraph of the Mean Intensities within the ROI

7.8.6 3D Mosaic

This command displays the ROI selection of an entire sequence in a single image. It does this by creating a new single-frame image and copying each frame of the original sequence to the next position in the new image. The command operates on the ROI of the active window and is scriptable. The active window must be a sequence containing any type of data.

3D Mosaic Example

For example, the above mosaic shows every second frame (frames #0, #2, #4, etc.) in the "Rhodamine Series" sequence.
The following dialog box produced this mosaic.

![3D Mosaic Dialog Box]

This command provides the following options:

**Scale to fit in monitor:** This option scales the mosaic image so that it fits on the main screen.

**Custom:** This option scales the mosaic image according to your setting. Use the associated menu and text boxes to enter the scale and the number of columns and rows to use.

If you select a scale of 1, the command creates a mosaic where each frame is shown at its actual size. A scale value of 2 shows each frame at half size. A scale value of 3 shows each frame at one-third size, etc.

The number of columns and rows determines the number of frames to include in each row of the mosaic and the number of rows to include in the final image.

**Include every n Frames:** Use this value to decrease the number of frames shown in the mosaic image. If you enter a value of 1, all frames are included. If you enter a value of 2, every other frame is included. If you enter a value of 3, every third frame is included, etc.

**Add Border:** When this option is selected, the command creates a black or white border for each frame. Use the associated Black and White radio buttons to specify the color.

**Add Labels:** When this option is selected, the command creates an overlay that shows the frame number in the lower left-hand corner of each frame.

To relocate the numbers, you can cut and paste this overlay using buttons in the Tools palette. You can also use the Change Segment Color command in the Analyze menu to display the numbers in a different color.
7.8.7 3D Oblique View

This command creates a new image that provides a pseudo-3D view of the image data within the ROI of the active window. It does this by displacing each pixel in the original image a distance that is proportional to the pixel value, then stacking the results. The active window may contain any type of data. Indexed type images should use the standard grayscale CLUT for best results. This command is scriptable.

**3D Oblique View Dialog Box**

**Darkfield:** When this option is selected, pixel displacement is directly proportional to pixel value. This mode works well with images where the objects of interest are brighter than the background (fluorescence or dark field microscopy, for example).

**Brightfield:** When this option is selected, pixel displacement is inversely proportional to pixel value. This mode works best in the case of a dark object against a brighter background.

**Projection factor:** Enter the projection factor as a percentage. The projection factor determines how much the image is compressed vertically. At 100%, the image is uncompressed. At the minimum value of 10%, the image is maximally compressed. Changing the projection factor can give the illusion of altering the viewing angle. At 100%, the image is viewed from above. At 10%, the image is viewed from the bottom edge.

**Stacking depth:** Enter the stacking depth in pixels.

**Stacking angle:** Enter the stacking angle (or direction) in degrees. At 90 degrees, the image is stacked to the right. At -90 degrees, the image is stacked to the left. At 0 degrees, the image is stacked straight ahead.

**Do All Frames:** When checked, the command will produce a new sequence by applying the stacking to each frame of an existing sequence. When unchecked, the command only operates on the current frame of the source image.
### 3D Stacked View

This command can be used to synthesize the data contained in a sequence into a single 2D image. One of four different mathematical operations are performed on a sequence. The original image is left unchanged, and the result is displayed in a new, singled framed, image window. The command can be applied to images of type Byte, Short Integer, Unsigned Integer, and Color 24. The resulting window is of the same type as the original sequence, except when you select **Sum**. The **Sum** option can not be used on color images, and always places the results in a Long Integer image.

#### 3D Stacked View Dialog Box

**Max (Darkfield):** The value at each pixel (x,y) location is assigned the highest value for that location across all frames of the sequence. This option is useful when observing dark field images, such as fluorescently labeled samples. In these images, objects of interest are brighter than the surrounding background.

**Min (Brightfield):** The value at each pixel (x,y) location is assigned the lowest value for that location across all frames of the sequence. This option is useful when observing bright field images, where the objects of interest are darker than the surrounding background.

**Average:** The result at each pixel location is the average of all frames at the same position. Use this option to improve the signal to noise ratio of an image series.

**Sum:** The result at each pixel location is the sum of all frames at the same position. Use this option to improve the signal to noise ratio of an image series, and to obtain the total pixel values for the entire sequence.
7.8.9  3D Tracer

This command provides a convenient way of viewing, slicing, measuring, and tracing curves and surfaces on your 3D data. It creates three orthogonal views of your 3D sequence, letting you view three perpendicular planes at once. You can record the position and intensity or color value at any point in 3-dimensional space and draw separately on each frame of the sequence. If the views are too large to fit on the main screen, the size of the image is reduced so that all views fit on the screen. The command operates on the ROI of the active window and is not scriptable. The active window must be a sequence that contains Byte or Color 24 data.

**3D Tracer Dialog Box**

**Interpolated frames:** Enter the number of frames to insert between each frame in the original sequence. You may want to add interpolated frames in order to increase the depth of the image. Enter a value between zero and 50.

To determine the appropriate value, you should acquire an image of an object with a known size, such as a ruler or a grid. If the distance between pixels in the X or Y dimension is smaller than the distance between acquired planes, you need to enter an interpolation value. For example, assume you have a distance of 0.4 microns between pixels in the images, and a distance of 2.0 microns between optical slices. Therefore, the recommended number of interpolated frames is:

\[(2.0 / 0.4) - 1 = 4.\]

**Duplicate:** When this option is selected, interpolated frames are merely duplicates of the preceding frame. If you are adding interpolated frames, this mode is faster but provides more uneven results.

**Linear:** When this option is selected, linear interpolation is used to calculate pixel values in interpolated frames. If you are adding interpolated frames, this mode is slower but provides smoother results.

Given values in the original sequence \( Seq[x, y, z] \), values in interpolated frames are determined by the following expression when the **Linear** option is used:

\[
value[x, y, t] = Seq[x, y, z] + \\
\frac{d \cdot (Seq[x, y, z+1] - Seq[x, y, z])}{n + 1}
\]
where:

\[ n \] is the number of interpolated frames

\[ d \] is the interpolated frame number (1 \( \leq d \leq n \))

\[ t = z \times n + d \]

Measurements:

**Meas. Window:** Enter the name of the window that will hold any measurements that you perform. If the window does not exist, it will be created once a measurement is made. If it already exists from a previous use of the Tracer command, the new measurement will be appended after the last non-zero row in the window.

**Maximum Number of Measurements:** Enter the maximum size of the measurements window. This size will be used when the measurements window is first created.

### 7.8.9.1 Example of Using the 3D Tracer

As an example, assume we have a ten-frame sequence, each frame containing a cross-sectional slice of a ring. We can use the 3D Tracer command on the sequence to create the images shown below.

![3D Tracer Example](image)

**3D Tracer Example**

Shown to the right of the three orthogonal views is the 3D Tracer window, which is similar to the **IPLab Tools** and **Status** palettes. The 3D Tracer window displays information about your sequence and provides a way to select tools.

The 3D Tracer window shows the cursor coordinates (in pixels), the value of the pixel under the cursor (a single value for Indexed type images, three values corresponding to red, green and blue for direct type images). The image width, height, and depth, the data type, and the magnification level are also given. The 3D Tracer command draws lines on the three views indicating which plane is shown in the other views. Under the Lines label, the 3D Tracer window shows the coordinates (in pixels) where the lines intersect and the pixel value at that point.
The tools in the top group on the 3D Tracer window (labeled Lines) allow you to change the cutting planes in the 3D data set, as shown by the colored lines drawn on the orthogonal views. The three cross-hair tools let you change cutting plane positioning. First click on the tool then click the cursor inside the image. The tool in the first row changes both the horizontal and vertical cutting planes; the left tool in the second row only changes the horizontal cutting plane; and the right tool in the second row only changes the vertical cutting plane. As you change the cutting plane in one view, a different image plane is displayed in one of the other views.

You can also use the arrow keys to change cutting plane positions as described below:

**Left Arrow:** Moves the cutting plane in the negative X direction.

**Right Arrow:** Moves the cutting plane in the positive X direction.

**Up Arrow:** Moves the cutting plane in the negative Y direction.

**Down Arrow:** Moves the cutting plane in the positive Y direction.

**Cmd-Up Arrow or Cmd-Left Arrow:** Moves the cutting plane in the negative Z direction.

**Cmd-Down Arrow or Cmd-Right Arrow:** Moves the cutting plane in the positive Z direction.

The **Hide** button allows you to hide the cutting plane lines drawn on the orthogonal views. After you hide the lines, the button is re-labeled **Show**. Clicking the **Show** button restores the cutting plane lines. Below the **Hide** button is a pop-up menu that can be used to change the line color.

Clicking the **Record** button records the coordinates where the cutting plane lines intersect and the pixel value at that point. These data are stored in a window named **Measurements**. If this window does not already exist, it is created behind the orthogonal views. If the window does already exist, then the measurements are appended to the list. This window can hold up to 100 measurements. You can also record measurements by typing **Command-R**.

The second group of tools on the 3D Tracer window (labeled Draw) is for drawing on the orthogonal views. Before you can start drawing, you must click the **Start** button. This creates a new drawing sequence that is used to store the information drawn on each frame. The drawing tools included on the 3D Tracer window are the **Paintbrush** tool, the **Eraser** tool, the **Line** tool, and the **Polygon** tool. These tools operate identically to the tools provided on the **IPLab Status** palette (see the Tools section of the **IPLab** manual). As you draw in one view, another view may update if the drawing intersects the cutting plane displayed in that view. Below the **Start** button is a pop-up menu that can be used to change the drawing color.

The check boxes in the 3D Tracer window determine what happens when you close the 3D Tracer window by clicking on the **Done** button. These options are described below:

**Keep views:** When this box is checked, the orthogonal views are kept.

**Keep drawing:** When this box is checked, the drawing sequence (if it exists) is kept.

**Stamp Drawing on Original:** When this box is checked, the information in the drawing sequence (if it exists) is stamped onto the original sequence. You can use this option to draw annotations on the original image.

This option is only available if the number of interpolated frames is zero.
7.8.10  3D Projector

This command creates a new sequence of 3D projections through the data set. The projections are done at a series of angles around either the X or Y axis. The command is scriptable.

The command operates on the ROI selection of the active window. The active window may contain Byte or Color 24 data.

Projection processing is a very computer intensive task, and the projection of a large image sequence (Width x Height x Number of frames) can take some time. Projection of color images takes longer since all colors are treated independently. You should consider selecting a rectangular region of interest of the smallest possible size that includes the object of interest before using this command.

As an example, the 3D Projector command is applied to the ten-frame ring sequence shown before to create the projection sequence shown below.

![3D Projector Example](image)

### 3D Projector Dialog Box

- **Generate Projection Sequence**
  - Rotation Axis: X axis, Y axis
  - Imaging Mode: Darkfield, Brightfield
  - Starting Angle: 0
  - End Angle: 360
  - Increment Angle: 15
  - Interpolate Frames in Z: 2
  - Use Distance modulation: 100
  - Anti-aliasing Filter (slower)
  - Disable display

![3D Projector Dialog Box](image)
This command provides the following options:

**X axis:**
When this option is selected, the X axis is used as the rotation axis.

**Y Axis:**
When this option is selected, the Y axis is used as the rotation axis.

**Darkfield:**
With this option, the projection is optimized for images that have brighter objects against a darker background (fluorescence or darkfield microscopy images, for example).

**Brightfield:**
With this option, the projection is optimized for images that have dark objects against a brighter background (transmitted light images, for example).

**Starting Angle:**
Enter the starting angle in degrees for the projection. This should be a value between zero and 360.

**End Angle:**
Enter the ending angle in degrees for the projection. This should be a value between zero and 360.

**Increment Angle:**
Enter the increment angle in degrees for the projection. This should be a value between 5 and 359.

**Interpolate Frames in Z:**
Enter the number of frames to insert between each frame in the original sequence. Interpolated frames are merely duplicates of the preceding frame. You may want to add interpolated frames in order to increase the depth of the image. (For more information, see the description of Interpolated frames for the 3D Tracer command.) Enter a value between zero and 50.

**Use Distance Modulation:**
When this option is selected, objects in the projection are dimmed in direct proportion to their depth. This feature provides depth cueing. If you are using this option, specify the amount of distance modulation in the associated text box. Enter a value of 1 for minimum distance modulation and a value of 100 for maximum distance modulation. See the algorithm explanation below for more details.

**Anti-Aliasing Filter:**
When this option is selected, an anti-aliasing filter is applied to the projection. In this mode all interpolated frames are used in the calculations, and filtering is applied to smooth the projection at steep angles. This option provides slower but smoother results.

**Disable display:**
When this option is selected, the new image is not updated every time a projection frame is created. You may want to use this option to speed up your results.

The new sequence has the name of the original window plus the following letters:

**X** or **Y:** To indicate the axis of rotation

**D** or **B:** To indicate either Darkfield or Brightfield mode.

### About Imaging Mode

The Darkfield option includes darkfield and fluorescence microscope operating modes. In the case of Darkfield microscopy, an object has a brightness related to its light scattering properties (refractive index variations). In the case of fluorescence microscopy, an object with uniform loading has a brightness related to its thickness. In the rendering scheme used for this option, the object itself is considered to be the light source, and the maximum value encountered in the process of Ray Casting is displayed. You may optionally elect to reduce the intensity of pixels which are farther form the viewer. This is called Distance Modulation and enhances the apparent 3D nature of the projected view.
The Brightfield option includes Transmitted light, Phase Contrast, and Differential Interference Contrast (DIC) microscope operating modes. In each of these cases, objects of interest appear darker than the background media. In the rendering scheme used for this option, the minimum value encountered in the process of Ray Casting is displayed.

Note: The option of Distance Modulation is not available for brightfield images. If you wish to apply the Distance Modulation option to a Brightfield image, you may want to first apply the Invert command to the image (Enhance menu) to convert it into a darkfield image, then use this command with the Darkfield option.

7.8.10.2 Description of Projection Algorithm

A new sequence is created to contain the projections. The number of frames depends on the angle settings entered in the dialog, and it is equal to:

\[
Frames = \frac{(\Theta_{\text{end}} - \Theta_{\text{start}})}{\Theta_{\text{inc}}}
\]

Each frame of the Projection sequence contains a view at a different angle \( \Theta \). The axis of rotation is always at the center of the Region of Interest.

For each of the projection angles, a number of parallel rays (one for every pixel) are cast and penetrate through the volume. The view from a particular angle is built in stages as every Z frame from the original sequence is considered. The ray casting continues until all the volume voxels have been used.

For example, at angle Theta (\( \Theta \)), and position \((x,y)\), the following formulas relate the position of a voxel in 3D space to its position in the projection plane (2D space). The number of interpolations used in the rendering is represented by \( Int \).

\[
\begin{align*}
X' &= x \\
y' &= y \cos(\Theta) + (z \cdot Int) \cdot \sin(\Theta) \\
z' &= -y \sin(Q) + (z \cdot Int) \cdot \cos(\Theta)
\end{align*}
\]

The \( x \) and \( y \) coordinates \((x',y')\) are used to determine the position on the projection plane. The depth, or \( z' \) position is used for the Distance Modulation option. The intensity of a voxel using modulation \( I_{\text{mod}} \) depends on its distance to the viewer, and decreases as the square of the distance until it reaches its lowest value. The lowest value is a percentage of the original voxel value determined by the \% Modulation value entered in the dialog.

Menu Reference - 3D

285
\[ I^{\text{mod}}(x', y') = \left( \frac{\text{DepthMax} - z'}{\text{DepthMax}} \right)^2 \cdot \left( 1 - \frac{\text{Mod}\%}{100} \right) \cdot I(x, y, z) \]

where:

\( \text{DepthMax} \) is the maximum distance possible.

\( \text{Mod}\% \) is the value entered for **Distance Modulation \%** within the dialog. The points farthest from the viewer have their intensity decreased by this percentage value. For example, at \( \text{Mod}\% = 75 \), a solid object with an intensity of 200 counts will have its lowest value equal to 50 counts.

At every \( \Theta \) setting the intensity of every voxel is projected, intensity modulated, and accumulated. In the case of a Darkfield image the maximum value encountered is kept for each XY position. In the case of a Brightfield image the minimum value encountered is kept for each XY position. In the case of Color 24 data, each color channel is processed independently.

### 7.8.11 Sample 3D Scripts

3D comes with a collection of scripts as an example of some of the tasks that can be performed automatically. You will find these in the "Scanalytics Scripts" folder:

IPLab 3.6 Folder: IPLab 3.6 User Folder: IPLab Scripts: Scanalytics Scripts

You can select the different scripts with the **Open Script** command (Script menu), and then press the **Run Script** button to proceed with the operation. You can also use the command **Assign F Keys** (Edit menu) to run a script with a single keystroke.
The scripts included are:

Make Stereo Frame: This script applies to the active window, which must be a grayscale sequence of Z sections. Using some of the commands of 3D, such as 3D Projector, this script creates a single color stereo view. You should use red (left) & green (right) glasses to observe the image.

Make Stereo Sequence: This script applies to the active window, which must be a grayscale sequence of Z sections. Using some of the commands of 3D, such as 3D Projector, this script creates a color stereo sequence. You should use red (left) & green (right) glasses to observe the animated image.

Make Stereo Pair: This script applies to the active window, which must be a grayscale or color sequence of Z sections. In this case the script creates a pair of images side by side.

You can edit these scripts by adding your own commands to perform a particular task.
7.9 Camera Menu

The Camera menu contains the commands for acquiring images from your camera.

Please read the Camera Menu Commands chapter of the Extensions manual, in Part I: Acquisition. It describes each of the Camera menu commands in detail.

7.10 Control Menu

The Control menu contains all commands that control microscope hardware. These commands are device independent, insofar as they and their dialog boxes remain the same regardless of the identity of the hardware.

You will find the Control menu commands described in the Control Menu Commands chapter of the Extensions manual, in Part II: Motion Control.
7.11 Script Menu

You can switch to any open script by picking its name from the list at the bottom of this menu.

7.11.1 New Script

This command creates a new, empty script window. Please read the Scripting chapter (page 135) for a full description of scripts.

7.11.2 Open Script

This command prompts you for the script to be opened.

![Open Script Dialog Box](image)

When you click on the **Open Script** button, the highlighted script is read from the disk and placed in a new script window. *IPLab* always opens scripts with recording turned off, to reduce the chance of accidentally adding commands to an existing script.

This command cannot be scripted.
7.11.3  Run Script

This command prompts you for the script to be run.

![Run Script Dialog Box]

**Run Script** Dialog Box

When using **Run Script**, no script window will be created unless the script is stopped by an alert, an error, or by the user pressing the keys Command-Period or Escape. In those cases, the script window will appear. If an error stopped the script, the command that gave the error is highlighted. Otherwise, the next command to execute is highlighted so that you may continue the script with the **Continue** button.

The **Run Script** command may be placed into a script. This lets you make master scripts with secondary scripts that work like subroutines. Running one script from another lets you make shorter, simpler scripts. However, there are certain limitations which you should be aware of first. For more information, see the section in the Scripting chapter titled “Running One Script from Another,” on page 149.
7.11.4 Set Variable

The **Set Variable** command assigns a value to an **IPLab** variable. Since many commands can take their parameter values from variables, it helps to have a convenient way to set and modify the values of those variables. While you could select parts of the **Variables** window and enter values with the **Set ROI Value** or **Paste** commands, the **Set Variable** command is much more convenient and versatile.

**Set Variable** Dialog Box

First, in the **Variable #** box, enter the number of the variable (i.e. the index, between 0 and 255) whose value you want to set.

The **Operation** section provides several options for setting the variable’s value. The first, **Set To**, simply sets the value to the indicated value. Notice that you can set the value of one variable equal to the value of another variable with this option.

The second option allows you to do some arithmetic on two variables or on a variable and a constant. You can
- add,
- subtract,
- multiply,
- divide,
- find the maximum or minimum, or
- multiply by a square root.

When you use the **MAX** or **MIN** functions, the variable will be set to the maximum or minimum of the two values. When you use the **√** function, the first value will be multiplied by the square root of the second value. If you just want the square root of a number, enter the number one as the first value, pick **√**, and enter the number in question as the second value.

The **Use Pixel Value At** option is good for setting a variable to the value of a pixel (or data cell) in another image.
- First, enter the coordinates of the desired pixel value in the **X** and **Y** boxes.
- Then tell the dialog box if the coordinates are in the active window or in another window, which you can pick or type in the **Window** box.

Although the **Variables** window is of Floating Point data type, you can use the **Set Variables: Use Pixel Value At** command to take the value from a data window of any type.

This feature is good for copying values from measurement results tables to variables. Often, such an operation would precede performing arithmetic on the measurement result.

The **Use Pixel Value At** option is good for setting a variable to the value of a pixel (or data cell) in another image.

Each of the numeric parameters in the **Set Variable** dialog can also be obtained from a variable by toggling the **#** to **Var**.
7.11.5  Action Variables

The Action Variables command lets you dynamically set the values of variable while a script is running. You also design your own interface for setting variables. This command gives you even greater interactive control over IPLab's behavior than ever before.

7.11.5.1  The Dialog Box

The Action Variables dialog box lets you set up the interface for active setting of variables, which is the floating palette:

First, choose the Action Variables command from the script menu. Then fill in the Action Variables dialog box. Each of the dialog's numeric values can also be obtained from a variable by toggling a # button to Var.

Window Name: Give the floating palette a name. If you are modifying an existing palette, choose its name by clicking on the pop-up list button .

Checkbox: Each checkbox enables a new control interface. Click in a checkbox to turn on a control. Then click the blue arrow (so it points down) to see the settings for this control.

Label: Give the control a name; it will be displayed within the floating palette.

Var #: Enter the number (the index) for the variable whose value you want to set.

Display: Choose the Display tab when you want to show the current value for a variable. If your script is placing a measurement into a variable, as in our example dialog box shown above, then you can use the Display tab to give you a live report of that measurement.
Buttons: Present two or three custom buttons to the IPLab user by filing in the Buttons tab. Enter the button's text in the Name fields. Show button #3 by checking its checkbox. For the Var # field, choose a variable to store the ID of the last button pressed.

When the user clicks on your buttons, the variable will be set to the value 0, 1, or 2, depending on which button the user presses. Your script (which is running the Action Variables command) can then use If statements to tell the script what to do next.

The Buttons tab is similar to the Query command found on the Script Commands palette.

Slider: Users can set the variable's value by sliding the handle back and forth across the slider. They can also change the value up or down by one by clicking on the spinner.

Type: The slider can set integer or floating point values.

Min, Max: These are the limits on values that the slider can set.

# of Ticks: This is the number of tick marks displayed under the slider. Click Sticky? to make the slider's handle stick to the tick marks. Then the user will only be able to slide to values at the tick marks.

Cancel: Click Cancel to close the Action Variables dialog box without creating or changing a palette.

Dispose: Click Dispose to close the Action Variables palette named in the Window Name field. Record this Action Variables: Dispose command late in the script, when you no longer need a floating palette to be open.

Create/Change: This button initially says Create, and says Change when the Action Variables palette already exists. Click this button to make a new Action Variables palette, or to apply the changes you just made to the parameters.

Record this command at the start of a script, or when you want the Action Variables palette to appear. If you record this command within a loop, the floating palette may flicker. However, you may want to record this command in a script if the Action Variables parameters (such as Number of Ticks) are being set dynamically, such as by another Action Variables command.

7.11.5.2 Using Action Variables

Use the Action Variables command by recording it in a script. You can use Action Variables outside of a script, but all it will do is set the value(s) of one or more variables. In a script, you can use Action Variables to exert live control over IPLab's behavior.
7.11.6 Continue

When an alert or other interruption has suspended a script, the Continue command becomes active. Selecting Continue will run the script beginning with the currently selected command.

7.11.7 Cancel All

When an alert or other interruption has suspended a script, the Cancel All command becomes active. Selecting Cancel All stops the script.
7.12 Window Menu

7.12.1 Show/Hide Tools

This command shows the Tools palette if it is not currently visible, and hides it if it is already visible. Whenever the Tools palette is visible, it appears to float above all other windows, but it is never considered the active window.

When you script this command, a dialog asks if the window is to be shown or hidden:

![Show/Hide Tool Palette Dialog Box]

7.12.2 Show/Hide FKeys

This command shows the FKeys palette if it is not currently visible, and hides it if it is already visible. Whenever the FKeys palette is visible, it appears to float above all other windows, but it is never considered the active window.

When you add this command to a script, a dialog asks if the window is to be shown or hidden:

![Show/Hide FKey Palette Dialog Box]

This command will be grayed out and read, "No FKeys" if no commands are assigned to function keys. Use the Assign F Keys command (Edit menu) to assign commands and scripts to function keys.

7.12.3 Show/Hide Status

This command shows the Status palette if it is not currently visible, and hides it if it is already visible. Whenever the Status palette is visible, it appears to float above all other windows, but it is never considered the active window.

You can also show and hide the Status palette by pressing the Command and U keys at the same time (Command-U).
When you add this command to a script, a dialog allows you to choose whether the window is to be shown or hidden:

![Show/Hide Status Dialog Box](image)

**Show/Hide Status Dialog Box**

### 7.12.4 Show/Hide Variables

The **Variables** window is always available, but it may be hidden. Use this command to show the **Variables** window as the front window, or, if the window is already shown, to hide it.

When you add this command to a script, a dialog allows you to choose whether the window is to be shown or hidden:

![Show/Hide Variables Dialog Box](image)

**Show/Hide Variables Dialog Box**

### 7.12.5 Dispose Window

**Dispose Window** closes the active window without asking if you want to save any changes. It is important to remember that, unlike the **File** menu’s **Close** command, **Dispose Window** will not prompt you to save the data even if you’ve made changes since the last time you saved. Be careful with this command, since it cannot be undone.

You can also dispose of windows by pressing the Command and K keys together (Command-K).

When you add this command to a script, a dialog will ask you which window to dispose. You may dispose the front window, or any other named window:

![Dispose Window Dialog Box](image)

**Dispose Window Dialog Box**

### 7.12.6 Dispose All Windows

This command closes *all* data windows. It is the same as using the **Dispose Window** command on every open window. It is scriptable, but does not have a dialog box. Be careful with this command, since it cannot be undone!
7.12.7 Rename Window

This command’s dialog prompts you for a new name for the active window. In addition, you may choose to start with the existing name, or use a new name, and then append various details to that name.

You can call this command by pressing the key combination Command and R (Command-R).

Use Existing Name: Click this if you want to append information to the window’s current name.

Replace With: You can use this to simply replace the current name, or you can also append information using the following options.

To append information to the window name, click one of the following checkboxes. Each of these items will be preceded by a space.

Text: Enter any additional text you like. When scripting, this is very useful for adding identification tags to existing names.

Date: The date format is set by the Preferences command (Edit menu).

Time: The time format is set by Preferences, as well.

Number: You can append up to ten digits to the name. That number can be obtained from the IPLab Variables if you toggle the # button to show Var.

Zero Filled: If you check the Zero Filled option, numbers below 1000 will be written in the form 0xxx. This form is compatible with the indexed file names.

IPLab allows you to use any characters in the window name. However, Macintosh file names may not contain the colon (:) character, and if you are saving to another file system, other characters may also be disallowed. In the case of date and time formats that include colons, the colons are changed to semicolons (;) before appending to the name. In addition, Macintosh file names are limited to 31 characters.

If you are renaming a window that has already been saved to a file, the old file will not be affected. If you save the renamed file, it will be placed in the same folder as the original file.
7.12.8  Duplicate Window

This command creates a new window containing some or all of the selected window’s data.

You can duplicate the front window or any open window.

You can duplicate All Frames within an image sequence, or only a range of the frames. (If you are duplicating a single-frame image, it does not make a difference.) When using Frames $a$ through $b$, the starting and/or ending frame numbers may optionally be obtained from the IPLab variables.

The Portion options let you duplicate either the entire image or just the part within the region of interest.

You may enter a name for the new window or accept the default name, by leaving the name blank. The default name is the original name with & appended to the end.

To crop an image, first select the desired portion with an ROI tools, Then use the Duplicate Window command with Portion set to ROI. This creates a new image that contains only the portion you selected from the original. You can also use the Crop Image command (Edit menu), which is more basic.

7.12.9  Move Window

This command lets you move a window around the screen. It is most useful in scripts where you may want to move one or two windows to see them better without performing a Tile or Stack command on all of the open windows. The dialog lets you specify whether to move the window to a specific location (Absolute) or by a certain amount from its existing location (Relative). The origin, (0,0), is at the upper-left corner of the screen, and distances are positive moving down and to the right.
7.12.10 Change Window

This command changes which window is the active window.

![Change Window Dialog Box](image)

There are three options for making a window active:

Using the **Named** option, you can either type the window name or select the name from the list of currently open windows by clicking on the arrow button. The windows are listed from top to bottom in the pop-up list in the order they appear on the screen, with the top window as the active window.

The **Move Next Window Up** and **Move Bottom Window to Top** options offer an alternate way to make a window active. Using these options becomes easier if you think of the list of windows as a circular list or stack. When you use **Move Next Window Up**, the currently active window goes to the bottom of the stack, the second window in the stack becomes the active window, and every other window in the stack is moved upward. This is an upward shift of the window stack. When you use **Move Bottom Window to Top**, the window at the bottom of the stack becomes the active window, and every other window in the stack is pushed downward. This is a downward shift of the window stack. You cannot bring a script window to the front using **Move Bottom Window to Top**.

The **Move** options are particularly useful in scripts when you may not know the name of the window, but you do know, for example, that it is immediately below the last window that you created. It can be important to keep track of where windows are within the window stack.

When making a script, **Change Window** allows you to enter any window name into the **Named** field, whether or not the window exists at the time you are editing the script. That way, a running script can activate a window that is created earlier in the script.

7.12.11 Frames to Sequence

With this command, you can create an image sequence out of individual image windows or files. There must be enough memory to hold the entire image sequence when it is created. Any windows or files that contain more than 1 frame will not be included in the new sequence.

If you want to get the individual frames from files, you must specify either a file list or the base name for **Indexed Files**.
Frames to Sequence Dialog Box

7.12.12 Sequence to Frames

With this command, you can separate an image sequence into individual frames. The individual frames are created with names of the form “base0000, base0001...”, which is the form of indexed files. This command will always save the frames to disk as individual image files. The individual frames will open on screen, as well, if there is enough memory free to hold all of them.

Click on the Output Format pop-up menu to set the format of the new files. The Same as Sequence option will write the individual frames in the same file format as that of the original sequence. You can also choose from IPLab, TIFF, PCIT, FITS, or Raw (the EPR format).

To set the base name and file path, you must run Set Index Name by clicking on the button here or by choosing the command from the File menu before splitting the sequence. When scripting Sequence to Frames, you should include the Set Indexed Name command earlier in the script.

Sequence to Frames Dialog Box
7.12.13  **Tile**

*Tile* distributes the open windows evenly around the screen by shrinking the windows and placing them in rows and columns. This command is scriptable.

You can also tile windows by pressing the Command and semicolon keys (⌘;).

7.12.14  **Stack**

*Stack* arranges the open windows in a cascading fashion from the screen’s upper left to its bottom. This command is scriptable.

7.12.15  **Window List**

A list of the open windows follows the *Window* menu commands. Selecting one of the window names from this list makes it active.
8 Writing Extensions

8.1 Introduction

You can easily extend the capabilities of IPLab to meet your own special needs by incorporating functions and routines built from your own code. These additions, called extensions, are effectively new IPLab commands that can be used the same way as resident IPLab commands. Your extensions can use dialogs to request parameters from the user, and they can be used in scripts.

To develop IPLab extensions, you need three things: a development environment capable of creating stand-alone code resources, the information in this manual and in the IPLab sample extensions, and a rudimentary knowledge of Macintosh programming.

You use a development environment to create your code, usually in a high level language. Since IPLab extensions are just specially constructed code resources, you can probably develop them in almost any development environment. We supply example code and documentation for the Metrowerks’ CodeWarrior Pro 4 (PowerPC only).

The sample extensions provided with IPLab consist of header and interface files, example source code, and the project files needed to compile and link each sample into an extension.

8.2 Architecture of IPLab Extensions

This chapter describes the development of extensions for version 3.6 of IPLab. The header file (IPLabExtensions.h) has defined the symbol IPLAB35HEADERS to identify new features added since version 3.2. The symbol is defined by default, to encourage the development of extensions that are compatible with IPLab versions 3.5 and 3.6. You can remove the definition to make the header file compatible with existing extensions.

IPLab continues to support older extensions (back to version 2.5), but you should try to conform to the 3.5 version of the interface. The About IPLab command provides a dialog box for each extension command that lists some useful information for each extension, including its compatibility with versions 3.5 and 3.6.

8.2.1 IPLX Files

Extensions are contained in files that are separate from IPLab. A Macintosh file has two designations that can be accessed by other programs to determine what type of file it is without opening the file: the file type, and the file creator. To create extensions that work properly with IPLab, they must reside in files of type ‘IPLX’. When you launch IPLab, the program looks for files of type IPLX in its own folder. If it sees one or more such files, it assumes each of these files contains IPLab extensions and it looks for specific resources within these files. The file creator information is not used by IPLab, but if the creator is set to ‘IPLB’, then the Finder assigns it an IPLab icon to show that the file contains IPLab extensions.
Each extension within an IPLX file consists of the following resources:

- A code resource which executes your function.
- An information resource which tells IPLab about your extension.
- A resource which lists the menu name of each command in your extension.
- Additional, custom, resources that contain information used by the code, including dialogs and other information for communicating with the user.

Each extension should reside in its own file.

8.2.1.1 The IPMX Code Resource

The code, of course, is essential and IPLab recognizes resources of type ‘IPMX’ as code resources within IPLX files. Each IPMX resource must have a unique resource number. The IPMX code resource allows a single extension to handle more than one command. Each IPLX file should contain only one IPMX resource. The IPLX file must also contain an IPmn resource (with the same resource ID as the IPMX resource) that specifies the command names that will be used in IPLab menus and scripts. See the section “The IPmn Menu Resource”, below, for more information.

It is best to mark the IPMX resource as both “Preload” and “Locked”. IPLab will work with purgeable extension resources, however their use can introduce memory fragmentation problems. Marking the resource as preloaded and locked causes the code to be loaded into the heap when IPLab starts up, and never moved afterwards.

8.2.1.2 The IP25 Information Resource

The ‘IP25’ resource was introduced with version 2.5 and its presence indicates that your extension follows the interface described in this manual. Version 3.0 and later of IPLab requires that your extension have an IP25 resource. The resource ID must be 128. This resource tells IPLab about your extension before the extension is actually opened. The IPLabExtensions.r file provides a template for this resource.

The resource is a single four byte value, broken into bit fields:
The first bit identifies the type of code: 0 (zero) indicates 68K code and 1 (one) indicates PowerPC code.
The second bit identifies whether the extension is compatible with the window library introduced in version 3.5 of IPLab. 0 (zero) indicates that it is not, and 1 (one) indicates that it is. This bit only applies to PowerPC extensions.

The above rules result in three possible values for this resource: 0 for 68K extensions, 1 for PowerPC extensions that do not use the IPLab Window Library, and 3 for PowerPC extensions that do use the IPLab Window Library.
8.2.1.3 The IPmn Menu Resource

To associate the command names that are used in *IPLab* menus and scripts with the entry codes understood by the IPMX resource, an IPmn resource must be included in your IPLX file. The IPmn resource must have the same resource ID as the IPMX resource. All of the Sample Extensions have been built to use IPMX and IPmn resources. The IPLabExtensions.r file provides a template for the IPmn resource.

An example in Rez format is shown below:

```rez
resource 'IPmn' (20080, purgeable)
{
    mExtensions,
    {
        20080, 10, "Sample 08 Fill ROI?",
        20080, 20, "Sample 08 Fill Non-ROI",
    }
};
```

The first number (20080) is the resource ID of the IPMX resource. The next number is the command ID that you have assigned to the command. There are no particular values for this number; *IPLab* will pass it to you when it calls your extension. The third and final item for each command is the name that will appear in the Ext. menu, and in scripts.

8.2.1.4 Other Resources

Your extensions may use other resources such as dialogs, alerts and private resources. Different extensions may even share resources. These resources are not required in order to run your extensions, but dialogs are especially convenient if you want your users to communicate parameter values to your extensions. Each resource used by *IPLab* must have a unique resource ID number. Scanalytics has created a resource numbering scheme that will guarantee that the resources you create for your extensions will not conflict with other resources that are used by *IPLab*. Your extensions may use any type of resources you want as long as you follow the resource ID guidelines given below.

8.2.1.5 Extension Resource Numbering

In order to avoid conflicts among resources, all resources used in your extensions, including the IPMX resources themselves, must have resource IDs numbered according to the following rules:

<table>
<thead>
<tr>
<th></th>
<th>For Scanalytics Inc.</th>
<th>For Third Party Developers</th>
<th>For Your Own Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10,000</td>
<td>10,000 – 19,999</td>
<td>20,000 – 29,999</td>
<td></td>
</tr>
</tbody>
</table>

By adhering to these guidelines, you can assure that your internal extensions will not conflict with extensions produced by Scanalytics or third-party developers (i.e. other than Scanalytics). To avoid conflicts among extensions produced by third-party developers, those developers should register their resource ID numbers with Scanalytics. The sample extensions use resource ID numbers above 20,000. You may want to begin each new project with a copy of one of the sample extensions and modify it to your own needs.

8.2.2 Communicating with IPLab

Some information in this section is derived from Macintosh Operating System data structures. These structures are described through references to one of the first five volumes of *Inside Macintosh*, the primary Macintosh programmer’s guide. The notation “IM V-48”, for example, refers to volume V, page 48. However, you should not need to refer to the *Inside Macintosh* volumes unless you want to include dialogs or similar interface elements in your extensions.
8.2.2.1 The Extension Parameter Block

 Extensions communicate with *IPLab* through a “parameter block”. The parameter block is a struct of type TExtensionRecord, described below. The address of the parameter block is passed from *IPLab* to your extension on the stack when the extension is called. It is the only parameter to your extension.

The ordering of the fields and definitions for each field in the TExtensionRecord follow:

<table>
<thead>
<tr>
<th>Field Names:</th>
<th>C Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fAction</td>
<td>short</td>
</tr>
<tr>
<td>fErr</td>
<td>short</td>
</tr>
<tr>
<td>fVarDataPtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fParameters</td>
<td>Ptr</td>
</tr>
<tr>
<td>fParameterSize</td>
<td>short</td>
</tr>
<tr>
<td>fEntryCode</td>
<td>short</td>
</tr>
<tr>
<td>fProcList</td>
<td>TCallBackPtr</td>
</tr>
<tr>
<td>fPBSize</td>
<td>long</td>
</tr>
<tr>
<td>fRefCon</td>
<td>long</td>
</tr>
<tr>
<td>fIPLabVers</td>
<td>unsigned long</td>
</tr>
<tr>
<td>fExtResID</td>
<td>short</td>
</tr>
<tr>
<td>fExtResFileRef</td>
<td>short</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>TExtensionRecord;</td>
</tr>
</tbody>
</table>

In describing each field, the notation > is used to indicate that *IPLab* provides information to your extension through this parameter, while the notation < indicates that *IPLab* expects your extension to provide information to it through this parameter. The notation (><) indicates that information is passed both ways through this parameter. Constants are used in several of the field definitions. The values of these constants are declared in the appropriate header files on the distribution disk.

**fAction (>)**

This field tells the extension what to do. Possible values are:

- **kInitAction**: Sent to each extension when *IPLab* is first started. Use this call to perform any one-time initialization that you may need.
- **kDialogAction**: Present the user with a dialog to get any parameters needed for the command to execute.
- **kDoAction**: Execute the extension’s function.
- **kQuitAction**: Sent to each extension just before *IPLab* quits. Use this call to release system resources, close hardware drivers, etc.
- **kScInquireAction**: This will be sent the first time your extension is placed into a script. Your response indicates whether your dialog routine may be called while the script is executing (more below, under “Scripting”).

Generally, *IPLab* expects that when kDialogAction is passed to the extension, the extension will do only its dialog (or nothing), not the actual function, and when kDoAction is passed, the extension will do only its function, not present a dialog.

**Interactive Operation**: When the user selects your extension from the menu it will actually be called twice: the first time with fAction = kDialogAction, then, if no error is returned by the first call, with fAction = kDoAction. If your extension does not use a dialog it should return kNoErr in the fErr field when it is called with fAction = kDialogAction.
**Scripting:** When a user selects your extension from the menu while a script is being made, fAction is set to kDialogAction. If you don’t return an error, your extension will be included as a step in the script. When that step is reached while the script is executing, fAction is set to kDoAction. If you return an error the script will stop.

In *IPLab* version 3 and later, scripts allow the dialog to be displayed during execution of the script. Of course this makes no sense if your extension doesn’t have a dialog. To find out if this is the case, *IPLab* will call your extension with fAction set to kScInquireAction when the user first places your extension into a script. You return a 1 (one) in the first byte of fParameters if you can display your dialog during script execution. Return a 0 (zero), or just leave fParameters untouched if you cannot. This call will be made after the kDialogAction call. *IPLab* remembers this value until the user quits, so the kScInquireAction call will only be made once during any run of *IPLab*.

Extensions do not have to do anything during Init or Quit calls, but these are good places to restore and save your dialog parameters to resources, so the user will see the same values the next time *IPLab* is run.

Changes to this field are ignored.

fErr (<)
This is a return parameter to be set by the extension before returning to *IPLab*:
- kNoErr no error encountered
- kExtensionErr error during execution
- kUserCancelled user canceled dialog

If fAction was set to kDialogAction and your extension returns kNoErr then *IPLab* assumes the user accepted the dialog parameters and *IPLab* either inserts the command into the script or calls your extension again with fAction set to kDoAction. If fAction was set to kDialogAction and your extension returns kUserCancelled then *IPLab* assumes the user rejected the dialog parameters and *IPLab* does nothing. When *IPLab* is running a script and encounters your extension, it will be called with fAction set to kDoAction. If your extension returns kExtensionErr then *IPLab* will stop the script.

fVarDataPtr (>)
This is the address of the *IPLab* Variables. You can use the Variables as a scratch pad to pass information back and forth between *IPLab* and your extension. Although a typical Macintosh user interface requests parameter values from the user through a dialog, dialogs can be difficult to program. If you do not want to bother with dialogs, just have the user set specific elements in the Variables Window. Then you can read these values in your extension using this field. The Variables array is also a handy way to pass information between different extensions. Remember that *IPLab* Variables is a 1D data array of single-precision floating point values, so, the \( n \)th element begins at byte fVarDataPtr + 4* \( n \) ( \( n \) goes from 0 to 255).

**Your extension must not change the value of this field. Only the data it points to may be modified.**

fParameters (><)
This is the address of the space used to hold an extension’s dialog parameters. When your extension is called with fAction = kDialogAction, you may use a dialog to get parameters from users, place them in the space pointed to by fParameters, and change the fParameterSize field described below. When your extension is called with fAction = kDoAction, it should use the values pointed to by fParameters. This area holds a maximum of 2272 bytes. You may use as little or as much of this space as you’d like. You may use any types of parameters you wish, including numbers, strings or other. *IPLab* does not try to interpret or change the bytes held in your dialog parameters.

When *IPLab* calls your extension with fAction = kScInquireAction (see fAction field description, above) the fParameters field can only be assumed to point to a single byte of data with a value of zero.

**Your extension must not change the value of this field. Only the data it points to may be modified.**

fParameterSize (<)
You must set this field to the actual size, in bytes, of the parameters that you have placed in the fParameters field. This field is initialized to zero when your extension is called, so if you do not use parameters, leave this field set to zero, and *IPLab* will know that there are no parameters to store.
fEntryCode

**IPLab** will pass a command entry code whenever your extension is called with fAction = kDialogAction, kDoAction, or kScInquireAction. This code corresponds to the one listed in the IPmn resource for the extension. It identifies the command that is being called.

fProcList

This is the address of a list of callback procedure pointers. The list is indexed by constants defined in the header files. Future callbacks will be defined by appending them to this list. See “Using Callbacks”, below, for a complete list of available callbacks.

fPBSize

This is the size of the parameter block itself, in bytes. When making copies of the parameter block you can use this field to determine the space needed for the copy.

fRefCon

This is a 4 byte field reserved for your use. **IPLab** saves the value of this field between calls to your extension. See the section “Making use of the fRefCon Field” for more information.

fIPLabVers

This is a 4 byte field that holds the version of **IPLab** that is calling your extension. It is built from the values in **IPLab**’s ‘vers’ resource. From high byte to low they are: Major, Minor, Interim, and Stage. The first two are in BCD format. The interim letter is coded as a number with ‘a’ = 1. The last field will be 0x20 (development), 0x40 (alpha), 0x60 (beta), or 0x80 (release). Two examples:

<table>
<thead>
<tr>
<th>Version</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.0 no interim</td>
<td>0x02500080</td>
</tr>
<tr>
<td>2.5.g ‘final’</td>
<td>0x02510780</td>
</tr>
</tbody>
</table>

In most cases you should only use the upper two bytes to check which version of **IPLab** is running.

fExtResID

This is the resource ID of your extension’s IPMX resource.

fExtResFileRef

This is the Resource file reference number for your extension. It is the number used in calls like UseResFile and UpdateResFile. This makes it easier to add resources to your extension file (see “Saving Data Between Launches”).

308 Writing Extensions
8.2.2.2 The Image Reference Block

To work with IPLab windows from your extension, you need to use an Image Reference parameter block. This struct contains dynamically allocated fields (a region and a polygon) that raises the specter of memory management. IPLab provides callbacks to initialize, copy, and release these records, to help reduce the code you need to write. For more information, see the section “Image Reference Callbacks”, below.

<table>
<thead>
<tr>
<th>Field Names</th>
<th>C Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef struct {</td>
<td></td>
</tr>
<tr>
<td>fWPtr</td>
<td>WindowPtr</td>
</tr>
<tr>
<td>fWindowKind</td>
<td>short</td>
</tr>
<tr>
<td>fDataType</td>
<td>DataTypes</td>
</tr>
<tr>
<td>fWidth</td>
<td>long</td>
</tr>
<tr>
<td>fHeight</td>
<td>long</td>
</tr>
<tr>
<td>fNumFrames</td>
<td>short</td>
</tr>
<tr>
<td>fCurrentFrame</td>
<td>short</td>
</tr>
<tr>
<td>fDataPtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fFramePtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fDisplayPtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fCTablePtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fROILeft</td>
<td>long</td>
</tr>
<tr>
<td>fROITop</td>
<td>long</td>
</tr>
<tr>
<td>fROIRight</td>
<td>long</td>
</tr>
<tr>
<td>fROIRegion</td>
<td>RgnHandle</td>
</tr>
<tr>
<td>fROIPolygon</td>
<td>PolyHandle</td>
</tr>
<tr>
<td>fDataMinPtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fDataMaxPtr</td>
<td>Ptr</td>
</tr>
<tr>
<td>fOverlay</td>
<td>PixMapHandle</td>
</tr>
<tr>
<td>fDataChanged</td>
<td>short</td>
</tr>
<tr>
<td>fReNormalize</td>
<td>short</td>
</tr>
<tr>
<td>fCLUTChanged</td>
<td>short</td>
</tr>
<tr>
<td>fRecSize</td>
<td>short</td>
</tr>
<tr>
<td>} TImageReference;</td>
<td></td>
</tr>
</tbody>
</table>

**fWPtr (>)**
This parameter points to a Macintosh window. The WindowPtr type is defined in IM I-275. It is possible for fWPtr to be Nil, when, for example, there is no front window.

A Macintosh WindowPtr points to a record. Most of the fields in the window record are not useful to you. If you want to get the window’s title, you should use the Macintosh Toolbox call GetWTitle. In C it is defined as: void GetWTitle (WindowPtr theWindow, Str255 title).

**You must not change this parameter in any way.**

**fWindowKind (>)**
This parameter indicates what kind of window fWPtr is:

- kImageWindowKind The active window is an image window.
- kNoWindow There is no active image window.

Changes to this parameter are ignored.
fDataType (>)
This parameter indicates the type of data displayed in the window:

- ByteImage: each pixel is a byte (8 bits)
- ShortImage: each pixel is a word (16 bits)
- LongImage: each pixel is two words (32 bits)
- FloatImage: each pixel is a floating point value (32 bits)
- Color 24Image: each pixel is 32 bits of RGB color, formatted as 8 bits unused, 8 bits Red, 8 bits Green, 8 bits Blue
- UnsShortImage: each pixel is an unsigned word (16 bits)
- Color 48Image: each pixel is 48 bits of RGB color, formatted as 16 bits each Red, Green, and Blue

Changes to this parameter are ignored. Your extension cannot change the Data Type of any existing IPLab window.

fWidth (>)
The width of the image, in pixels. Changes to this parameter are ignored. Your extension must use the CBAlterImageDimensions callback to change the dimensions of any existing IPLab window.

fHeight (>)
The height of the image, in pixels. Changes to this parameter are ignored. Your extension must use the CBAlterImageDimensions callback to change the dimensions of any existing IPLab window.

fNumFrames (>)
The number of frames held in this window. Changes to this parameter are ignored. Your extension must use the CBAlterImageDimensions callback to change the dimensions of any existing IPLab window.

fCurrentFrame (>)
This is the zero-based index of the current frame in the window (i.e. the displayed frame). It indicates the number of the frame to which the fFramePtr field is pointing. Changes to this parameter are ignored.

fDataPtr (>)
This parameter contains the address of the entire sequence of frames for a window. It is a true Macintosh heap pointer. You can use this address, in conjunction with the fDataType, fWidth, fHeight, and fNumFrames fields, to calculate the address of any pixel in any frame. The address of the current frame in the user’s window is passed to you in fFramePtr (see below).

Your extension must not change the value of this field. Only the data it points to may be modified.

fFramePtr (>)
This parameter contains the address of the start of the current frame of image data. This address is used with the pixel size (fDataType) and the width and height to access the data in the current frame.
This field is simply an address, and you should not pass it to Macintosh OS calls such as GetPtrSize.

Examples:
- pixel (10,20) in a byte image of size 100 x 200 is at byte fFramePtr + 10 + 20*100
- pixel (5,30) in a short integer image of size 100 x 200 starts at byte fFramePtr + 2*(5 + 30*100)
- the Green component of pixel (5,30) in a Color 24 image of size (100,200) is at byte location fFramePtr + 4*(5 + 30*100) + 2.

Your extension must not change the value of this field. Only the data it points to may be modified.

fDisplayPtr (>)
This is a pointer to the byte or color array used to display the image data. Normally IPLab handles all of the computations necessary to compute the values stored in this array through the Normalization command. However, this field is provided for those who want to use the Normalized values, rather than the actual image values. For Color 24 and Color 48 images, the display data type is Color 24. For all other data types the display data type is Byte.
If you fill this buffer, you should also set fReNormalize to kDontReNormalize to tell IPLab not to renormalize (When IPLab performs normalization, it overwrites this buffer.).

Your extension must not change the value of this field. Only the data it points to may be modified.

fCTablePtr (>)
This is a pointer to the Color Table used to display the image. The Color Table itself is a cSpecArray of size 256 as described in IM V-136. You may consider it an array of 1024 short integers organized as follows:

- index 0, red, green, blue
- index 1, red, green, blue
  ...
- index 255, red, green, blue

At present, the indexes must be consecutively number from 0 to 255. If you change the color table be sure to adhere to this rule, and be sure to alert IPLab that you have changed the table by setting the fCLUTChanged parameter (see below).

Your extension must not change the value of this field. Only the data it points to may be modified.

fROI.left, fROITop, fROIRight, fROIBottom (<>)
These parameters are the pixel coordinates of the rectangle which bounds the Region Of Interest in the window. If the ROI is a rectangle, then these parameters define the ROI. If you change these parameters, the changes are applied to the window when you call the UpdateImage callback, but will not be seen until you return to IPLab. If you change these parameters to make a rectangular ROI, you should not change fROIPolygon!

fROIRegion (>)
This parameter is a handle to a region describing the ROI in the window. The structure of a region is not well documented by Apple, however a general description of it is given in IM I-141. It is primarily useful to test whether a specific point is within a non-rectangular ROI defined by the user. The example code on the distribution disk shows how this is done using a Macintosh Toolbox routine called PtInRgn. You may access this field, but you must not change this field in any way!

fROIPolygon (>)
This parameter is a handle to the polygon which describes the corner points of the ROI. This field is easier to use than the fROIRegion field if all you want are the corner points. The structure of a polygon is described in IM I-159. You may access this field and change it to make a new non-rectangular ROI. The changes will be applied to the window when you call the UpdateImage callback, but will not be seen until you return to IPLab. If you change this field you must know how to work with polygons and you should not change fROI.left, fROITop, fROIRight, or fROIBottom!

fDataMinPtr, fDataMaxPtr (<>)
Use these pointers to tell IPLab what the new minimum and maximum values over the entire image are. Normally, you should let IPLab compute these. But if you are writing into the array pointed to by fDisPtr, for example, to do your own display normalization, and you set fReNormalize to kDontReNormalize, then you must compute these values. If the Data Type is Color 24 or Color 48 then these values are never used. They are passed in as zero, and any changes to them by your extension are ignored.

IPLab treats the data addressed by these pointers as variables of the same type (and size) as the image data itself. You must re-cast these pointers to examine or change the min. and max. values. For example, if fDataType = longIntImage, then your C code should look something like this:

```c
long computedMinValue;
...
*(long*)fDataMinPtr = computedMinValue;
```

Your extension must not change the value of these fields. Only the data they point to may be modified.
fOverlay (>)
This field holds a handle to a pixmap that contains the overlay image. A pixmap is a Macintosh data structure, and it is described in *Inside Macintosh*. The overlay data is held in 4 bits per pixel, and the values for the colors used are listed in the header files for *IPLab* extensions. The easiest way to change the overlay is to draw into it using QuickDraw commands. There are callbacks provided (see below) that will assist you in setting up the drawing environment for overlays.

fDataChanged (<)
This is a return field. Set it to 1 to tell *IPLab* that the extension has changed the data pointed to by fDataPtr. Then *IPLab* will ask you if you want to save the image when you close the window or quit *IPLab*. Otherwise, set it to zero.

fReNormalize (<)
This is a return field. Set it to:

- kDontReNormalize: If you do not want *IPLab* to perform normalization.
- kForceReNormalize: If you definitely want *IPLab* to re-normalize the image when it next displays.
- kQuickReNormalize: If you want to normalize using the existing table (even if data has changed).
- kNoNormChange: otherwise (*IPLab* automatically normalizes data in newly created windows).

Important: If you change the data values and want to set this field to kDontReNormalize to tell *IPLab* not to perform normalization, you must also compute the minimum and maximum pixel values over the image and change the values pointed to by fDataMinPtr and fDataMaxPtr as described above. *IPLab* usually recomputes the min. and max. values when it does renormalization. If you change the data values and tell *IPLab* not to renormalize, the fDataMinPtr and fDataMaxPtr values will be wrong unless you update them yourself.

fCLUTChanged (<)
This is a return field. Set it to 1 to tell *IPLab* that the extension has changed the CLUT so that *IPLab* will update the window and take this change into account. Otherwise, set it to zero.

fRecSize (>)
This is the size of the TImageReference record, in bytes. You should not need to use this field. Use the CopyImageRef callback to duplicate the record.

8.2.3 Using Callbacks

Callbacks are routines built into *IPLab* that you can call from your extension. In order for you to use these routines, you must provide them with the arguments they need. In this section, the arguments for each Callback are explained. We present a prototype for each Callback to indicate the order and type of each argument. The actual way you invoke Callbacks depends on the programming language and development environment you are using. The details for C are given in the next section.

The addresses of the callback functions are provided in an array. The index used to reference each callback is a constant name that is made up of ‘cb’ followed by the name of the routine. If your extension is native PPC code, a Universal Procedure Pointer (UPP) must be built for the callback. The C header file we supply contains macros that produce the UPP for you. Using the header declarations allow you to use a single macro which looks like a standard function call, and which expands to the proper code when compiled. The macro callback names all begin with ‘CB’ and all require the extension parameter block as the first parameter.

In describing each callback parameter, the notation > is used to indicate that the callback provides information to your extension through this parameter, while the notation < indicates that the callback expects your extension to provide information to it through this parameter. The first parameter, the extension parameter block pointer, is always required, and will not be further described.
8.2.3.1 Initializing and Releasing Image References

Any callback that operates on an *IPLab* window requires a pointer to an image reference parameter block (type TImageReference). This parameter block includes fields that allocate memory from the heap, (a region and a polygon). To manage the creation and disposal of these fields, routines have been declared to create, copy, and dispose of image references. Be sure to use these routines when working with image references.

**GetEmptyImageRef**

This takes a pointer to a TImageReference and initializes its fields. It allocates an empty region for the fROIRegion field. Use this to set up a reference for a new window that your extension will create.

Prototype:

```c
err = CBGetEmptyImageRef (TExtensionRecPtr p, TImageRefPtr imgRef)
```

err: (> ) The function returns kNoErr if the fields were initialized correctly.

imgRef: (> ) This is the address of a TImageReference record. It will be initialized by this call.

**GetFrontImageRef**

This takes a pointer to a TImageReference and sets its fields to match those of the front most *IPLab* image window. You should not use this function to determine if the front window is a script window. Use the OperatingMode callback instead.

Prototype:

```c
err = CBGetFrontImageRef (TExtensionRecPtr p, TImageRefPtr imgRef)
```

err: (> ) The function returns kNoErr if the fields were initialized correctly.

imgRef: (> ) This is the address of a TImageReference record. It will be initialized by this call to the front most window.

**CopyImageRef**

This copies an image reference into a second image reference record. It makes duplicates of the region and polygon fields.

Prototype:

```c
err = CBCopyImageRef ( TExtensionRecPtr p, TImageRefPtr sourceWndRef, TImageRefPtr destWndRef)
```

err: (> ) The function returns kNoErr if the fields were initialized correctly.

sourceWndRef: (< ) This is the address of a valid TImageReference record. It will be the source record.

destWndRef: (> ) This is the address of a valid (already initialized) TImageReference record. It will be the destination record.

**ReleaseImageRef**

This releases an image reference, and disposes of any memory allocated by GetEmptyImageRef or GetFrontImageRef. It has no return value.

Prototype:

```c
CBReleaseImageRef (TExtensionRecPtr p, TImageRefPtr imgRef)
```

imgRef: (< ) This is the address of a TImageReference record. It will be disposed of by freeing the region and polygon fields.
### 8.2.3.2 Working with Image Windows

#### NewImage

This function creates a new image window.

Prototype:

```c
err = CBNewImage ( TExtensionRecPtr p,
                   TImageRefPtr imgRef,
                   FileFormats theFormat,
                   DataTypes theType,
                   long width,
                   long height,
                   short numFrames,
                   ViewModes viewMode,
                   Str255 windowName,
                   void* dataPtr)
```

- **err:** (>) The function returns kNoErr if the new image was created. Insufficient memory, for example, prohibits the creation of a new window.

- **imgRef:** (>) This is the address of a TImageReference record. It must contain an initialized image reference. Upon return it will be set to refer to the new window.

- **theFormat** (<) This describes the default file format for this image. Acceptable values are:
  - IPLDImage
  - IPLab
  - PICTFormat
  - TIFFFormat
  - FITSFormat
  - TEXTFormat data values are saved as ASCII text
  - RawFormat the data is stored without a header of any kind
  - PICTQTFormat the data is stored with QuickTime compression

- **theType** (<) This is the type of data in the new image:
  - ByteImage each pixel is a byte (8 bits)
  - ShortImage each pixel is a word (16 bits)
  - LongImage each pixel is two words (32 bits)
  - FloatImage each pixel is a four byte (32 bits) floating-point value
  - Color 24Image each pixel is 4 bytes (32 bits), formatted as 8 bits unused, 8 bits Red, 8 bits Green, 8 bits Blue
  - UnsShortImage each pixel is a word (16 bits)
  - Color 48Image each pixel is 6 bytes, formatted as 16 bits Red, 16 bits Green, 16 bits Blue

- **width** (<) This is the width, in pixels, of the new image

- **height** (<) This is the height, in pixels, of the new image

- **numFrames** (<) This is the number of frames in the new image

- **viewMode** (<) This is the initial viewing mode for the new image:
  - viewAsImage display the image
  - viewAsText display the pixel values in text
  - viewAsQGraph display the data as a Q-Graph
  - viewPerspective display the data in perspective view
windowName

(<) This is a Pascal-style string of text. Pascal-style strings begin with a byte that gives the length of the string. The same number of bytes following the length are presumed to contain the text of the string. Since this is used as the default filename, the length of the name should be not more than 31 characters (the maximum length of a filename in the Mac operating system) and should not contain a colon (’:’).

dataPtr

(<) This is the address of the data that comprises the image. If you want an empty (all zero) image created, just pass Nil. If you allocate your own memory for dataPtr, make sure it is the right size. The data space must contain exactly:

\[ \text{width} \times \text{height} \times \text{numFrames} \times \text{typeSize} \text{ bytes} \]

where typeSize depends on the Data Type as described above in theType field. The address of the data in the new image is always returned in the fDataPtr field of imgRef.

**NewImage2**

This function is identical to the NewImage callback except that it can create a new image window without showing it.

Prototype:

```pascal
err = CBNewImage2 ( TExtensionRecPtr p, 
                   TImageRefPtr imgRef, 
                   FileFormats theFormat, 
                   DataTypes theType, 
                   long width, 
                   long height, 
                   short numFrames, 
                   ViewModes viewMode, 
                   Boolean showNow, 
                   Str255 windowName, 
                   void* dataPtr)
```

**showNow**

(<) If True, the Image Window is shown after it is created. If False, the window is hidden. This applies to the entire window, not just the contents.

**ShowImage**

This procedure tells IPLab to show the window referred to by imgRef. This is equivalent to the “ShowWindow” procedure in the Macintosh Toolbox, but it also updates internal parameters associated with the window and the display. This callback would normally be used to make an image window visible after it was created using the NewImage2 callback with the showNow parameter set to false. This procedure has no return value.

Prototype:

```pascal
CBShowImage (TExtensionRecPtr p, TImageRefPtr imgRef)
```

**imgRef**

(<) This is the address of a TImageReference record. It must contain an initialized image reference. It will not be changed by this routine.
DisposeImage
This procedure tells *IPLab* to dispose of the window referred to by imgRef. This is equivalent to the “Kill Window” command in *IPLab*, in that it does not ask if you want to save the image first. You should always use this routine to dispose of an image window, so that *IPLab* can properly dispose of related data structures to which you do not have access. This procedure has no return value.

Prototype:

```
CBDisposeImage (TExtensionRecPtr p, TImageRefPtr imgRef)
```

imgRef: (<) This is the address of a TImageReference record. It must contain an initialized image reference. It will be returned as an “empty” image reference.

UpdateImage
This procedure updates the window referred to by imgRef. It has no return value. Both the internal parameters associated with the window and the display are updated. Display normalization is performed if requested by the fRenormalize field of imgRef.

Example 1: Your extension gets information about the active window and changes some data values. Pass imgRef directly to UpdateImage to see the effects before leaving your extension.

Example 2: Your extension creates a new Image Window using NewImage then grabs image data from a special device and places it into memory starting at address fDataPtr. Pass the imgRef you got from NewImage to UpdateImage to see changes in the image data displayed live in your new window.

Example 3: Your extension gets the information about the active window from GetFrontImageRef into imgRef1. Your extension changes some data and allocates a new ImageReference variable, called imgRef2 that it uses to create a new data window for reporting results. You can call UpdateImage with imgRef1 to update the image display, then call UpdateImage with imgRef2 to update the window that is reporting results.

Example 4: Your extension cycles through all of the open windows using NextImage (see the description below) and sets the ROI to be the same in each. Your extension must call UpdateImage to actually change internal parameters associated with each window, including the ROI.

Prototype:

```
CBUpdateImage (TExtensionRecPtr p, TImageRefPtr imgRef)
```

imgRef: (<) This is the address of a TImageReference record. It must contain an initialized image reference. It will not be changed by this routine.

NextImage
This is a function that gets the information for the “next” Image Window in the window list. First, you should initialize an image reference with GetFrontImageRef. You should then pass this imgRef (or a copy of it) to NextImage. imgRef is then updated to give you the information about the Image Window below the Active window. If your extension changes imgRef to NextImage in a loop, each call to NextImage will successively step through the windows behind the front-most window. When NextImage is called after the last window, information on the front-most is returned, and a return value informs you that you have returned to the front window and are starting over again.

If there are no Image Windows at all, NextImage returns an error code indicating this.

Next Image returns information only about Image Windows, not script windows, or the Status palette. If the Variables window is showing, it is returned as one of the windows. If the Variables window is not showing it is not returned.

Remember that the ordering of the windows can be fairly arbitrary. It is based on the order in which they were originally opened, or the order in which the user subsequently clicked on them.

To get the name of the window, you should use the Macintosh Toolbox procedure GetWTitle().
Prototype:
\[
\text{err} = \text{CBNextImage (TExtensionRecPtr p, TImageRefPtr imgRef)}
\]
\text{err: } (>) \text{ This function returns one the following values: kNoErr no error, kNoWindows no windows to return kIsFrontWindow the window returned is the Front window }
\text{imgRef } (\Rightarrow) \text{ This is the address of a TImageReference record. When you call NextImage, imgRef must contain the information for one of the existing windows. On return, it contains information about the window below that window, or the front window if imgRef pointed to the last window on entry.}

**ShowHideDisplay**

This procedure tells *IPLab* to show or hide the image display of the window referred to by imgRef. This is equivalent to the “Show/Hide Display” command in *IPLab*. Hiding the display can free up memory by disposing of the display buffer and other memory used to build the display buffer. This procedure has no return value.

Prototype:
\[
\text{CBShowHideDisplay (TExtensionRecPtr p, TImageRefPtr imgRef, Boolean showIt)}
\]
\text{imgRef: } (<) \text{ This is the address of a TImageReference record. It must contain an initialized image reference.}
\text{showIt: } (>) \text{ True will show the display if it is hidden or do nothing if it is already shown. False will hide the display if it is shown, or do nothing if it is already hidden.}

8.2.3.3 Working with Labels and Overlays

**SetImageLabel**

This procedure allows you to set the label for any column or row in the window referred to by imgRef. The label is displayed when the image is viewed as text. This procedure has no return value.

Prototype:
\[
\text{CBSetImageLabel (TExtensionRecPtr p, TImageRefPtr imgRef, short index, Boolean setColumn, Str255 label)}
\]
\text{imgRef } (<) \text{ This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.}
\text{index } (<) \text{ This is the row or column in which to set the label. Remember that rows and columns are numbered starting at zero.}
\text{setColumn } (<) \text{ If True, the given column label is set. If False, the given row label is set.}
\text{label } (<) \text{ This is the address of a Pascal-style string of text. Pascal-style strings begin with a byte that gives the length of the string. That many bytes following the length are presumed to contain the text of the string. The maximum length of column labels is 49 characters.}
AddOverlay
This function adds a blank overlay to the window referred to by imgRef. If an overlay already exists for that window, *IPLab* does nothing. If there is insufficient memory to create the overlay, an error code is returned.

Prototype:
```c
err = CBAddOverlay (TExtensionRecPtr p, TImageRefPtr imgRef)
```

err: (>) The function returns kNoErr unless the overlay could not be created.

imgRef (<) This is the address of a TImageReference record. It contains information about the window you wish to operate on.

StartOverlayDraw
This function sets up drawing to the overlay in the window referred to by imgRef. You must call this routine before drawing into an overlay. After you call this routine, all calls to QuickDraw drawing routines, such as LineTo, FrameOval, *etc.*, result in drawing to the overlay until you call EndOverlayDraw.

Prototype:
```c
err = CBStartOverlayDrawPtr ( TExtensionRecPtr p,
TImageRefPtr imgRef,
short colorIndex,
TSaveDrawInfoPtr saveInfo)
```

err: (>) The function returns kNoErr unless an error prevented switching to the overlay.

imgRef (<) This is the address of a TImageReference record. It contains information about the window you wish to operate on.

colorIndex (<) This tells *IPLab* which overlay color you want to draw with. Acceptable values are:
- kOvlWhite
- kOvlRed
- kOvlYellow
- kOvlMagenta
- kOvlCyan
- kOvlBlue
- kTransparent
- kOvlSegGreen
- kOvlBlack

saveInfo (>) This is a pointer to a record that stores information about the graphics environment at the time before you call StartOverlayDraw. Save this pointer and pass it to EndOverlayDraw when you are finished drawing.

EndOverlayDraw
This procedure returns the drawing world to the condition it was in before you started drawing into the overlay. Call this when you are finished drawing. Be sure to pass the value of saveInfo from the last time you called StartOverlayDraw. This procedure has no return value.

Prototype:
```c
CBEndOverlayDraw (TExtensionRecPtr p, TSaveDrawInfoPtr saveInfo)
```

saveInfo (<) This is a pointer to a record that stores information about the graphics environment at the time before your last call to StartOverlayDraw. It is needed to restore the graphics environment.
SetOverlayColor

This procedure sets the color used by subsequent QuickDraw routines for drawing into the overlay in the window referred to by imgRef. It has no return value.

Prototype:

\[
\text{CBSetOverlayColor} ( \text{TEXTensionRecPtr } p, \text{TImageRefPtr } \text{imgRef, } \text{short } \text{colorIndex})
\]

\text{imgRef} (<) This is the address of a TImageReference record. It contains information about the window you wish to operate on.

\text{colorIndex} (<) This tells IPLab which overlay color you want to draw with. Acceptable values are:

- kOvlWhite
- kOvlRed
- kOvlYellow
- kOvlMagentan
- kOvlCyan
- kOvlBlue
- kTransparent
- kOvlSegGreen
- kOvlBlack

Note to users of version 2.5 and earlier: The IPLab Segmentation command can now draw using any of the overlay colors instead of just kOvlGreen (versions 2.3 and earlier) or kSegGreen (versions 2.4 to 2.5).

8.2.3.4 Working with Image Files

OpenImage

This function opens the specified file into a new Image Window. If the file cannot be opened, an error is returned.

Prototype:

\[
\text{err } = \text{CBOpenImage} ( \text{TEXTensionRecPtr } p, \text{TImageRefPtr } \text{imgRef, } \text{FSSpecPtr } \text{fsPtr, } \text{TFillChoice } \text{fileChoice, } \text{FileFormats } \text{fileFormat, } \text{Boolean } \text{forceOpenAs, } \text{Boolean } \text{showNow})
\]

\text{err:} (>\text{The function returns kNoErr unless the image cannot be opened.}}

\text{imgRef} (>\text{This is the address of a TImageReference record. It will be initialized by this call to refer to the new Image Window.}}

\text{fsPtr} (<\text{This is the address of an FSSpec that specifies where the image is located on disk. This file specification is only used if fileChoice is set to SingleFile.}}

\text{fileChoice} (<\text{This describes the method of targeting the disk file. Acceptable values are:}}

- SingleFile \text{open using name and path specified in fsPtr}
- ListedFile \text{open using name and path specified in current IPLab file list}
- IndexedFile \text{open using name and path specified in current IPLab indexed file}
fileFormat

(<=) This specifies the file format to use when opening this image. This information is only used if forceOpenAs is True. Otherwise, the format is determined by the file type. Acceptable values are:

- IPLDImage: IPLab native format
- PICTFormat: Macintosh PICT format
- TIFFFormat: TIFF format
- FITSFormat: FITS format
- TEXTFormat: data values are saved as ASCII text
- RawFormat: the data is stored without a header of any kind
- PICTQTF: the data is stored with QuickTime compression

forceOpenAs

(<=) If True, IPLab assumes that the data in the file is in the format specified by fileFormat. If False, the format is determined by the file type.

showNow

(<=) If True, the Image Window is shown after it is created. If False, the window is hidden.

SaveImage

This function saves the window referred to by imgRef. If the image cannot be saved, an error is returned.

Prototype:

```c
err = CBSaveImage ( TExtensionRecPtr p,
TImageRefPtr imgRef,
FSSpecPtr fsPtr,
TFileChoice fileChoice,
FileFormats fileFormat,
Boolean saveROI,
Boolean savePreview,
CodecInfoPtr codecInfoPtr)
```

err:

(>) The function returns kNoErr unless the image cannot be saved.

imgRef

(<=) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.

fsPtr

(<=) This is the address of an FSSpec that specifies where the image should be saved. This file specification is only used if fileChoice is set to SingleFile.

fileChoice

(<=) This describes the method of targeting the disk file. Acceptable values are:

- SingleFile: save using name and path specified in fsPtr
- ListedFile: save using name and path specified in current IPLab file list
- IndexedFile: save using name and path specified in current IPLab indexed file

fileFormat

(<=) This describes the format of data in a disk file. Acceptable values are:

- IPLDImage: IPLab native format
- PICTFormat: Macintosh PICT format
- TIFFFormat: TIFF format
- FITSFormat: FITS format
- TEXTFormat: data values are saved as ASCII text
- RawFormat: the data is stored without a header of any kind
- PICTQTF: the data is stored with QuickTime compression

saveROI

(<=) If True, the data within the ROI is copied to a new image window and this window is save to disk.

savePreview

(<=) If True, a small preview image is saved with the file.
codecInfoPtr  (<) This is the address of a CodecInfoRec that specifies codec settings. This information is only used if fileFormat is set to PICTQTFormat. Otherwise, this field can be set to Nil.

The CodecInfoRec is described below:

<table>
<thead>
<tr>
<th>Field Names</th>
<th>C Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>typedef</td>
<td>struct</td>
</tr>
<tr>
<td>theCodecType</td>
<td>CodecType</td>
</tr>
<tr>
<td>spatialQuality</td>
<td>CodecQ</td>
</tr>
<tr>
<td>depth</td>
<td>short</td>
</tr>
</tbody>
</table>

} CodecInfoRec;

codecInfoPtr

theCodecType This is the type of QuickTime codec, such as:
- 'rpza' video compressor
- 'jpeg' photo compressor
- 'rle' animation compressor
- 'raw' raw compressor
- 'smc' graphics compressor
- 'cvid' compact video compressor

spatialQuality This specifies the desired compressed image quality. Acceptable values range from 0 for minimum quality to 512 for normal quality to 1024 for lossless quality.

depth: This specifies the depth at which the image is likely to be viewed. If you set this parameter to 0, the QuickTime codec determines the appropriate depth for the source image. Values of 1, 2, 4, 8, 16, 24, and 32 indicate the number of bits per pixel for color images. Values of 33, 34, 36, and 40 indicate 1-bit, 2-bit, 4-bit, and 8-bit grayscale, respectively, for grayscale images.

GetImageFSSpec

This function returns the file specification (FSSpec) in fsPtr of the file associated with imgRef. If the image has not been saved, an error is returned.

Prototype:

```c
err = CBGetImageFSSpec ( TExtensionRecPtr p,
                         TImageRefPtr imgRef,
                         FSSpecPtr fsPtr)
```

err: (>)
- kNoErr for no error,
- kExtensionErr image not saved

imgRef  (<) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.

fsPtr  (>) This is the address of an existing FSSpec. IPLab returns the FSSpec of the image in the record pointed to by fsPtr.
8.2.3.5 Working with Image Data

SelectFrame
This procedure, which should only be applied to image sequences, selects a particular frame in the window referred to by imgRef. This is equivalent to the “Select Frame” command in IPLab. This procedure, however, does not change the displayed image. To do that you still need to call CBUpdateImage. If the frame number specified is less than zero or greater than or equal to fNumFrames, the procedure selects the nearest frame within the allowable range. This procedure has no return value.

Prototype:

```c
CBSelectFrame ( TExtensionRecPtr p, 
TImageRefPtr imgRef, 
short frameToSelect, 
Boolean calcMinMax)
```

**imgRef:** (<) This is the address of a TImageReference record. It must contain an initialized image reference. IPLab updates any fields of imgRef that might have changed by selecting a new frame.

**frameToSelect** (<) This is the new frame to select. Remember that frame numbering starts at zero.

**calcMinMax** (<) If True, IPLab determines new minimum and maximum values for the new frame and updates the fDataMinPtr and fDataMaxPtr fields in imgRef.

AlterImageDimensions
This procedure changes the dimensions of the window referred to by imgRef. If the new dimensions do not result in more data than the original dimensions, then the image is resized to the new dimensions. If the new dimensions result in more data, this procedure does nothing. After it resizes the data, it changes the displayed image. This procedure has no return value.

Prototype:

```c
CBAlterImageDimensions ( TExtensionRecPtr p, 
TImageRefPtr imgRef, 
long newWidth, 
long newHeight, 
short newFrames)
```

**imgRef:** (<) This is the address of a TImageReference record. It must contain an initialized image reference. IPLab updates any fields of imgRef that might have changed by altering the image dimensions.

**newWidth** (<) This is the new width of the image.

**newHeight** (<) This is the new height of the image.

**newFrames** (<) This is the new number of frames in the image.

ImgToWinCoord
This function converts a point in the window referred to by imgRef from image coordinates to window coordinates. Image and window coordinates may not be the same if an image is magnified or scrolled. This function does the conversion without checking that window coordinates are within the boundaries of the window or that image coordinates are within the boundaries of the image data. In addition, coordinates can only be converted if the window referred to by imgRef is viewed as an image. Otherwise, the function returns kExtensionErr.

Prototype:

```c
err = CBImgToWinCoord ( TExtensionRecPtr p, 
TImageRefPtr imgRef, 
Point iPoint, 
PointPtr wPointPtr)
```
err: (>) The function returns kNoErr if imgRef is viewed as an image.

imgRef: (<) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.

iPoint (<) This is a point in image coordinates.

wPointPtr (>) This is the address of an existing Point. IPLab converts iPoint to window coordinates and sets this point to the result.

WinToImgCoord
This procedure converts a point in the window referred to by imgRef from window coordinates to image coordinates. Image and window coordinates may not be the same if an image is magnified or scrolled. This function does the conversion without checking that window coordinates are within the boundaries of the window or that image coordinates are within the boundaries of the image data. In addition, coordinates can only be converted if the window referred to by imgRef is viewed as an image. Otherwise, the function returns kExtensionErr.

Prototype:

```
CBWinToImgCoord ( TExtensionRecPtr p, TImageRefPtr imgRef, Point wPoint, PointPtr iPointPtr)
```

imgRef: (<) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.

wPoint (<) This is a point in window coordinates.

iPointPtr (>) This is the address of an existing Point. IPLab converts wPoint to image coordinates and sets this point to the result.

UseImageOffset
This procedure gets or sets the image offset for the window referred to by imgRef. The image offset is given in image coordinates and is zero unless the image has been scrolled. A change to the image offset only changes the displayed data if the window is viewed as an image or as text. This procedure has no return value.

Prototype:

```
CBUseImageOffset( TExtensionRecPtr p, TImageRefPtr imgRef, PointPtr iPointPtr, Boolean get)
```

imgRef: (<) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.

iPointPtr (>>) This is the address of an existing Point. It receives or specifies the image offset.

get (<) If True, IPLab gets the image offset for the image and puts it into the Point pointed to by iPointPtr. If False, IPLab sets the image offset for the image using information in the Point pointed to by iPointPtr.
UseImageMag
This procedure gets or sets the image magnification for the window referred to by imgRef. The image magnification
given by magPtr is actually the log base 2 of the true magnification. Therefore, specify -2 for 1/4X magnification, 2
for 4X magnification, and zero for no magnification. A change to the image magnification only changes the
displayed data if the window is viewed as an image. This procedure has no return value.

Prototype:
   CBUseImageMag( TExtensionRecPtr p,
                  TImageRefPtr imgRef,
                  short* magPtr,
                  Boolean get)

imgRef:  (<) This is the address of a TImageReference record. It contains information about
         the window you want to change. The fields in imgRef are not changed by this
         routine.

magPtr   (><) This is the address of an existing Short Integer (16 bits). It receives or specifies
         the image magnification.

get      (<) If True, *PLab gets the image magnification and puts it into the variable pointed
to by magPtr. If False, *PLab sets the image magnification using information in the
variable pointed to by magPtr.

UseImageView
This procedure gets or sets the image viewing mode for the window referred to by imgRef. If the image cannot be
changed to the specified viewing mode, an error is returned.

Prototype:
   err = CBUseImageView( TExtensionRecPtr p,
                         TImageRefPtr imgRef,
                         ViewModesPtr viewModePtr,
                         Boolean get)

err:      (>) The function returns kNoErr unless the image cannot be changed to the specified
          viewing mode.

imgRef:   (<) This is the address of a TImageReference record. It contains information about
          the window you want to change. The fields in imgRef are not changed by this
          routine.

viewModePtr (><) This is the address of an existing ViewModes variable. It receives or specifies
             the image viewing mode for the image, which can be one of the following:
             viewAsData    display the image
             viewAsText    display the pixel values in text
             viewAsQGraph  display the data as a Q-Graph
             viewPerspective display the data in perspective view.

get       (<) If True, *PLab gets the image viewing mode for the image and puts it into the
          variable pointed to by viewModePtr. If False, *PLab sets the image viewing mode
          for the image using information in the variable pointed to by viewModePtr.
UseAcquireData
This procedure gets or sets the acquire data for the window referred to by imgRef. This information is used in the FITS header attached to *IPLab* images. It should be updated whenever an extension acquires a new image. This procedure has no return value.

Prototype:

```c
CBUseAcquireData ( TExtensionRecPtr p, 
                    TImageRefPtr imgRef, 
                    Ptr acqPtr, 
                    long acqSize, 
                    Boolean get)
```

- **imgRef** (<>) This is the address of a TImageReference record. It must contain an initialized image reference. The fields in imgRef are not changed by this routine.

- **acqPtr** (><) This is the address of an existing acquire data record. It receives or specifies the acquire data. The fields of this record are described below.

- **acqSize** (<) This is the size of the acquire data record in bytes.

- **get** (<) If True, *IPLab* gets information from the image and puts it into the record pointed to by acqPtr. If False, *IPLab* sets the acquire data for the image using information in the record pointed to by acqPtr.

The acquire data record is described below:

<table>
<thead>
<tr>
<th>Field Names:</th>
<th>C Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>typedef struct {</td>
</tr>
<tr>
<td>filledIn</td>
<td>Boolean</td>
</tr>
<tr>
<td>unused</td>
<td>Boolean</td>
</tr>
<tr>
<td>observeType</td>
<td>short</td>
</tr>
<tr>
<td>observeDT</td>
<td>unsigned long</td>
</tr>
<tr>
<td>exposureTime</td>
<td>long</td>
</tr>
<tr>
<td>gain</td>
<td>float</td>
</tr>
<tr>
<td>grabRect</td>
<td>Rect</td>
</tr>
<tr>
<td>binWidth</td>
<td>short</td>
</tr>
<tr>
<td>binHeight</td>
<td>short</td>
</tr>
<tr>
<td>} TAcquireDataRec;</td>
<td></td>
</tr>
</tbody>
</table>

- **filledIn**: If True, the fields of the record have been filled in.

- **unused**: Set to False.

- **observeType**: This is the type of image captured. Acceptable values are:
  - 0 for Object
  - 1 for Dark
  - 2 for Bias

- **observeDT**: This is the date and time of image capture. This is the value returned by the GetDateTime procedure in the Macintosh Toolbox.

- **exposureTime**: This is the exposure time in milliseconds used when the image was captured.

- **gain**: This is the gain used when the image was captured.

- **grabRect**: This is the rectangle used when the image was captured. The rectangle is relative to full-frame capture.
binWidth  This is the horizontal binning factor used when the image was captured.
binHeight  This is the vertical binning factor used when the image was captured.

**UseNormData**

This procedure gets or sets the normalization data for the window referred to by imgRef. This information is used to normalize *IPlab* images. Using this procedure to set new normalization data, however, does not change the displayed image. To do that you still need to call CBUpdateImage. This procedure has no return value.

Prototype:

```c
CBUseNormData ( TExtensionRecPtr p,
                 TImageRefPtr imgRef,
                 Ptr normPtr,
                 long normSize,
                 Boolean get)
```

- **imgRef** (<) This is the address of a TImageReference record. It must contain an initialized image reference. The fields in imgRef are not changed by this routine.
- **normPtr** (><) This is the address of an existing normalization data record. It receives or specifies the normalization data. The fields of this record are described below.
- **normSize** (<) This is the size of the normalization data record in bytes.
- **get** (<) If True, *IPlab* gets information from the image and puts it into the record pointed to by normPtr. If False, *IPlab* sets the normalization data for the image using information in the record pointed to by normPtr.

The normalization data record is described below, with two struct definitions:

```c
typedef struct {
    TDisplayRange gray,
    TDisplayRange red,
    TDisplayRange green,
    TDisplayRange blue
} TNormDataRecord;
```

- **gray**  The normalization values to use for all gray-scale images.
- **red, green, blue**  The normalization values to use for each channel of a Color 24 or Color 48 image.
typedef struct {
  TLabDouble blackPoint;
  TLabDouble whitePoint;
  TLabDouble gamma;
  ENormDataSource drFrom;
  UInt8 unused1;
  Uint16 unused2;
  Uint32 unused3;
} TDisplayRange;

blackPoint This is the minimum normalization value.

whitePoint This is the maximum normalization value.

gamma Gamma values range between 0.05 and 1.0. Use 1.0 for a linear response.

drFrom This is the source of normalization data. Acceptable values are:

  ndsUser Use the minimum and maximum values specified
  ndsFrame Use the minimum and maximum values for the current frame
  ndsSequence Use the minimum and maximum values for the entire sequence
  ndsSatFrame Maps 1% of brightest pixels to 255 and 1% of darkest pixels to 0
  ndsSatSeq Same as ndsSatFrame, but with the sequence’s minimum and
          maximum
  ndsROI Use the minimum and maximum within the ROI

unused1,2,3: Set to 0.

UseQGraphData

This procedure gets or sets the QGraph data for the window referred to by imgRef. This information is used in the QGraph view options for *IPLab* images. Using this procedure to set new QGraph data, however, does not change the displayed image. To do that you still need to call CBUpdateImage. Changes to QGraph data are only visible if the image is viewed as a QGraph. This procedure has no return value.

Prototype:

```c
CBUseQGraphData( TExtensionRecPtr p,
                 TImageRefPtr imgRef,
                 Ptr qGraphPtr,
                 long qGraphSize,
                 Boolean get)
```

imgRef (<> This is the address of a TImageReference record. It must contain an initialized image reference. The fields in imgRef are not changed by this routine.

qGraphPtr (->> This is the address of an existing QGraph data record. It receives or specifies the QGraph data. The fields of this record are described below.

qGraphSize (<> This is the size of the QGraph data record in bytes.

get (<< If True, *IPLab* gets information from the image and puts it into the record pointed to by qGraphPtr. If False, *IPLab* sets the QGraph data for the image using information in the record pointed to by qGraphPtr.)
The QGraph data record is described below:

<table>
<thead>
<tr>
<th>Field Names</th>
<th>C Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>gallery</td>
<td>TQGallery</td>
</tr>
<tr>
<td>plotType</td>
<td>TQPlotType</td>
</tr>
<tr>
<td>abscissaCol</td>
<td>short</td>
</tr>
<tr>
<td>yLine</td>
<td>GraphLineArray</td>
</tr>
<tr>
<td>xAxis</td>
<td>AccessFormatRecord</td>
</tr>
<tr>
<td>yAxis</td>
<td>AccessFormatRecord</td>
</tr>
<tr>
<td>gridColor</td>
<td>RGBColor</td>
</tr>
<tr>
<td>backColor</td>
<td>RGBColor</td>
</tr>
<tr>
<td>labels</td>
<td>PlotLabelArray</td>
</tr>
<tr>
<td>reserved1</td>
<td>Rect</td>
</tr>
<tr>
<td>reserved2</td>
<td>double</td>
</tr>
<tr>
<td>reserved3</td>
<td>double</td>
</tr>
<tr>
<td>reserved4</td>
<td>short</td>
</tr>
<tr>
<td>reserved5</td>
<td>short</td>
</tr>
<tr>
<td>reserved6</td>
<td>short</td>
</tr>
<tr>
<td>reserved7</td>
<td>long</td>
</tr>
<tr>
<td>reserved8</td>
<td>long</td>
</tr>
<tr>
<td>reserved9</td>
<td>long</td>
</tr>
<tr>
<td>reserved10</td>
<td>long</td>
</tr>
</tbody>
</table>

```c
typedef struct {
    TQGallery gallery;
    TQPlotType plotType;
    short abscissaCol;
    GraphLineArray yLine;
    AccessFormatRecord xAxis;
    AccessFormatRecord yAxis;
    RGBColor gridColor;
    RGBColor backColor;
    PlotLabelArray labels;
    Rect reserved1;
    double reserved2;
    double reserved3;
    short reserved4;
    short reserved5;
    short reserved6;
    long reserved7;
    long reserved8;
    long reserved9;
    long reserved10;
} QGraphData;
```

gallery: This is the type of graph. Acceptable values are:
LineGraph
BarGraph

plotType: This is the type of plot. Acceptable values are:
Linear
LogLinear
LinearLog
LogLog

abscissaCol: This is the column in the data that should be used for the x-axis. Set to -1 to plot against a linear x-axis.

yLine: This is an array of ten GraphLineRecords. The GraphLineRecord specifies settings, such as line color, line width, line pattern, etc., for each plotted line. The fields of this record are described below.

xAxis: This is the AccessFormatRecord for the x-axis. The AccessFormatRecord specifies settings, such as minimum value, maximum value, precision, etc., for each QGraph axis. The fields of this record are described below.

yAxis: This is the AccessFormatRecord for the y-axis. The AccessFormatRecord specifies settings, such as minimum value, maximum value, precision, etc., for each QGraph axis. The fields of this record are described below.

gridColor: This is the color of the QGraph grid.

backColor: This is the color of the QGraph background.
labels: This is an array of five PlotLabelRecords. The PlotLabelRecord specifies settings, such as font, font size, font style, etc., for each QGraph label. These records provide settings for the QGraph title, x-axis label, y-axis label, x-axis values, and y-axis values. The fields of the PlotLabelRecord are described below.

reserved1…10 These fields are reserved and should not be changed.

The GraphLineRecord is described below:

<table>
<thead>
<tr>
<th>Field Names</th>
<th>C Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>short</td>
</tr>
<tr>
<td>reserved1</td>
<td>SignedByte</td>
</tr>
<tr>
<td>linePat</td>
<td>TQLinePat</td>
</tr>
<tr>
<td>reserved2</td>
<td>SignedByte</td>
</tr>
<tr>
<td>lineWidth</td>
<td>TQLineWidth</td>
</tr>
<tr>
<td>token</td>
<td>short</td>
</tr>
<tr>
<td>lineColor</td>
<td>RGBColor</td>
</tr>
<tr>
<td>tokenColor</td>
<td>RGBColor</td>
</tr>
</tbody>
</table>

typedef struct {
    column short
    reserved1 SignedByte
    linePat TQLinePat
    reserved2 SignedByte
    lineWidth TQLineWidth
    token short
    lineColor RGBColor
    tokenColor RGBColor
} GraphLineRecord;

column This is the column in the data that should be plotted. Set to -1 to not plot data.

reserved1 This field is reserved and should not be changed.

linePat This is the line pattern. Acceptable values are:
        FullLine
        DashedLine

reserved2 This field is reserved and should not be changed.

lineWidth This is the line width. Acceptable values are:
        NoLine
        HalfPtLine
        OnePtLine
        OneHalfPtLine

token: This is the token shown with each data point. There are twenty possible tokens with values between 1 and 20. Set to 0 for no token.

lineColor This is the line color.

tokenColor This is the token color.
The AxisFormatRecord is described below:

```
typedef struct {
  TQAxisMode mode;
  TQAxisFormat format;
  short precision;
  Boolean reserved;
  Boolean useDataMinMax;
  double delta;
  double scaleMin;
  double scaleMax;
  RGBColor color;
} AxisFormatRecord;
```

- **mode**: This is the mode for tick marks. Acceptable values are:
  - `Axis10`: Use 10 ticks per axis
  - `ComputerBest`: Use number of ticks automatically selected by **IPLab**
  - `UserSelected`: Use delta between ticks specified by user

- **format**: This is the number format. Acceptable values are:
  - `FixFormat`
  - `SciFormat`

- **precision**: This is the number of decimal places.
- **reserved**: This field is reserved and should not be changed.
- **useDataMinMax**: If True, scaleMin and scaleMax are automatically set to the minimum and maximum data values.
- **delta**: This is the interval between tick marks. This value is only used if mode is set to `UserSelected`.
- **scaleMin**: This is the minimum axis value.
- **scaleMax**: This is the maximum axis value.
- **color**: This is the color of the axis.

The PlotLabelRecord is described below:

```
typedef struct {
  short fontID;
  short fontSize;
  Style fontStyle;
  RGBColor color;
  short reserved1;
  short reserved2;
  Rect reserved3;
  Str63 labelText;
} PlotLabelRecord;
```

- **fontID**: Short
- **fontSize**: Short
- **fontStyle**: Style
- **color**: RGBColor
- **reserved1**: Short
- **reserved2**: Short
- **reserved3**: Rect
- **labelText**: Str63

```
```
fontID          This is the font ID number.
fontSize       This is the font size.
fontStyle:      This is the font style.
color           This is the font color.
reserved1…3:    These fields are reserved and should not be changed.
labelText       This is the label text. Does not apply to x-axis or y-axis values.

**GetImageUnits**

This procedure returns information about the distance units associated with imgRef. It has no return value.

Prototype:

```c
CBGetImageUnits ( TExtensionRecPtr p,
                 TImageRefPtr imgRef,
                 TLabExtendPtr unitFactor,
                 Str255 unitName)
```

- **imgRef** (<) This is the address of a TImageReference record. It contains information about the window you want to change. The fields in imgRef are not changed by this routine.
- **unitFactor** (>) This is the address of an existing TLabExtend variable. In this parameter, *IPLab* returns the size of a single pixel in the image's distance units.
- **unitName** (>) This is the address of an existing string. In this parameter, *IPLab* returns the text string for the units. This string is the name that appears in the *IPLab Status* palette.

### 8.2.3.6 Miscellaneous Callbacks

**GetExtRefCon**

This function returns the fRefCon field that *IPLab* is storing for any currently loaded extension. This allows two or more extensions to share data that has been allocated by one of them. See the section entitled “Making Use of the fRefCon Field” for more information.

Prototype:

```c
refCon = CBGetExtRefCon (TExtensionRecPtr p, short resID)
```

- **resID** (<) This is the resource ID of the extension whose fRefCon field you want returned. If the requested extension is not loaded the function returns Nil.

**OperatingMode**

This function returns the current *IPLab* operating mode.

Prototype:

```c
mode = CBOperatingMode (TExtensionRecPtr p)
```

- **mode** (>) The function returns one of the following:
  - *OpModeNormal* interactive operating mode
  - *OpModeScriptEdit* Script editing operating mode
  - *OpModeScriptRunDialog* Script running with displayed dialog operating mode
  - *OpModeScriptRunDo* Script running without displayed dialog operating mode
  - *OpModeIgnore* reserved operating mode
ShowPalettes
This function shows or hides the *IPLab* Status and Tools palette windows. It also returns the previous shown/hidden state of the windows.

Prototype:

```
previous = CBShowPalettes (TExtensionRecPtr p, UInt8 shownPalettes)
```

- `previous` (> The function returns the previous shown and hidden states of the palette windows in the same format as shownPalettes.
- `shownPalettes` (<) A bit map representing the palette windows and the shown/hidden state that you want applied to those windows. The least significant bit represents the *Status* palette, and the next bit represents the Tool palette. Example: a value of 0 hides all palettes, and 3 shows both.

UpdatePalettes
This procedure updates both the *IPLab* Status and Tools palette windows. It has no return value.

Prototype:

```
CBUpdateStatusWindow (TExtensionRecPtr p)
```

8.2.4 Working with *IPLab*’s Floating Windows

*IPLab* 3.5 implemented floating windows (for example, the status and tool palettes) with the help of a library (the *IPLab* Window Manager Library) that keeps track of which windows float above others. For your extension to be compatible with *IPLab*, it must work with the same library and follow certain rules regarding windows.

If your extension does not use any dialogs, and does not make any Macintosh Window Manager calls, then you do not need to use this library.

8.2.4.1 Modal Dialogs

If your extension puts up modal dialogs, you need to precede the opening of any dialog with the call: `DeactivateFloatersAndFirstDocumentWindow()`. Once the dialog is dismissed you need to call `ActivateFloatersAndFirstDocumentWindow()`. These insure that the floating palettes are properly highlighted to show their state.

8.2.4.2 Window Manager calls

If your extension makes any Macintosh Window Manager calls, you need to replace some of them with alternative calls from the *IPLab* Window Manager Library. Use the following table as a guide:

<table>
<thead>
<tr>
<th>Mac Toolbox call</th>
<th><em>IPLab</em> Window Manager Library call</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectWindow</td>
<td>SelectExtWindow</td>
</tr>
<tr>
<td>FrontWindow</td>
<td>FrontDocumentWindow</td>
</tr>
<tr>
<td></td>
<td>(returns topmost non-floating, non-modal window)</td>
</tr>
<tr>
<td></td>
<td>- OR -</td>
</tr>
<tr>
<td></td>
<td>FrontExtNonFloatWindow</td>
</tr>
<tr>
<td></td>
<td>(returns topmost non-floating window, including</td>
</tr>
<tr>
<td></td>
<td>modal dialogs)</td>
</tr>
<tr>
<td>DragWindow</td>
<td>DragExtWindow</td>
</tr>
<tr>
<td>ShowWindow</td>
<td>ShowExtWindow</td>
</tr>
<tr>
<td>HideWindow</td>
<td>HideExtWindow</td>
</tr>
</tbody>
</table>

If you directly reference the windowKind field of a window record, you should instead use the library routines: `GetExtWindowKind` and `SetExtWindowKind`.  

332 Writing Extensions
If you directly reference the wRefCon field of a window record, you should instead use the library routines: GetExtWRefCon and SetExtWRefCon.

### 8.2.5 Using the Library

To access the IPLab Window Manager Library, you need to include the header file, IPLabWindowMgr.h in your source code. You also need to add the “IPLabWindowMgrLib stub” file to your project. This is a stub shared library that provides the names of the entry points, so that you can link your extension, but not the actual code. The actual library is built into the “IPLab Library” file that is installed along with IPLab. Your extension will use this same library, once IPLab loads your extension. The Sample Extension projects which include dialogs use IPLabWindowMgrLib.

### 8.2.6 Execution Sequence

When IPLab is launched (opened from the Finder, or by an Apple Event) it searches its directory for IPLX files. Each file of this type that is found is opened and, if it contains an IP25 resource, the extension's commands are added to the Ext menu or Control menu as specified in the file's IPmn resource. Each IPMX resource is then loaded and sent the kInitAction action message.

While IPLab is running, your extension may be sent kDialogAction and/or the kDoAction messages. While scripting, the extension may be sent the kScInquireAction message.

When the user quits IPLab, each extension is loaded and sent the kQuitAction message.

The order in which extensions are added to the menu, and sent the kInitAction and kQuitAction messages is determined by the order in which the IPLX files are found. That order is currently defined to be alphabetic, by the name of each extension file.

If the shift key is held down during IPLab startup, then IPLab never searches for IPLX files, never loads them, and never builds the Ext menu. If this has happened, the IPLab About box will include the message “Extensions Disabled”.

### 8.2.7 Making Use of the fRefCon Field

The fRefCon field in the Extension Parameter Block can be used to hold data that you need to share with other extensions. The fRefCon field is saved by IPLab between calls to your extension (but not between executions of IPLab).

The usual course of action is to allocate and initialize a pointer or handle during the Init action message, and pass this address back to IPLab in the fRefCon field. The size and format of this space is up to you, but remember that any memory you take leaves that much less for the user. If you don’t expect to change the amount of information held in this space, use a pointer. If you expect to change the size of the data you would be better off using a handle (or, use a pointer and make one of the fields in it a handle). Other extensions of yours can then access this address by using the GetExtRefCon callback. Sample extensions 05 and 07 provide example code for using the fRefCon field. During the Quit action message you should dispose of any memory you allocated during the Init action message. While this is not important now (since IPLab is quitting anyway), it makes your extension more robust, in the event we have some future reason to send a Quit message without leaving IPLab.

### 8.2.8 Saving Parameters Between Launches

IPLab saves the values of dialogs between launches, so users will likely expect your extension to do the same. One way to do this is to save your data in a resource attached to your extension. The format and type of resource is entirely up to you. IPLab helps you use such a resource by providing the Init and Quit actions, and your resource file ID. When you get the Init action message try to read in the resource and store its values in a pointer. Return this pointer to IPLab in the fRefCon field. When you get the Quit action message, change the data into a resource handle and write it to your extension file. Sample extensions 05 and 07 provide example code for performing these two steps.
8.2.9 Development Tips

**Memory Allocation:** It is your responsibility to keep track of all memory allocated by your extension. If you allocate space in the heap, and do not pass it back to **IPLab** as part of a new window, then you must either keep track of that space (so you do not re-allocate it), or you must dispose of it before returning to **IPLab**. Otherwise, each call to your extension will cause a block of memory to be allocated and then orphaned, resulting in a steady decline in available memory. You should always use the allocation routines provided by the Macintosh Operating System (such as NewPtr, NewHandle, etc.), in preference to the standard routines provided by the various languages (such as malloc in C, or New in Pascal).

**Types of Native code containers:** There are two ways in which native PPC code can be packaged for use as a code resource. One is as an “accelerated” resource, the other is a “naked” native resource. **IPLab** version 3 supports both of these, however, for future compatibility, you should only create “naked” resources.

Accelerated resources are designed to be called just like 68K code resources. The resource contains a native code fragment. The first instruction, however, is a 68K trap instruction which causes the system to prepare and execute the rest of the native code fragment automatically. There is one important problem that can occur with accelerated resources: if the extension requires a shared library that has not been installed on the machine, the caller (**IPLab**) will crash with a system error. This is due to the way accelerated resources are prepared by the system.

“Naked” resources contain only a native code fragment. They must be prepared and executed by the caller (**IPLab**). Since **IPLab** will do the preparation itself, it can catch the error, and give a warning message instead of crash. Therefore it is better to build naked resources. Future versions of IPLab may make this a requirement.

**68K FPU code:** When running on a Power Macintosh, **IPLab** will load both 68K and PPC extensions (as determined by the value of the IP25 resource). Since the PowerMac contains a 68K emulator, the 68K extensions will continue to work. However, the emulator does not include emulation for the 68K FPU (floating point unit). If a 68K FPU instruction is executed by the emulator, **IPLab** will crash with a system error. There is no way for **IPLab** to know if an extension contains FPU instructions, so you must check for this yourself if you want to avoid this crash. The best way is to use the Mac’s “Gestalt” call during the kInitAction call to see if an FPU exists. If your code requires an FPU, and there is none, then return an error and **IPLab** will not load your extension.

8.3 Development Environments

This section gives a detailed description of how to create **IPLab** extensions using the most common Macintosh C development environment: CodeWarrior from Metrowerks. The standard **IPLab** installation includes folders that contain sample projects. Specifically, there are PPC projects for CodeWarrior Pro, Release 4.

The sample projects consist of header and interface files, example source code, and any specific commands or information needed to compile and link a sample into an extension.

Other languages and development environments may also be used to create IPLX files. **IPLab** extensions do not depend on the development environment or language you use. Use the information presented here and the sample extensions on the **IPLab** distribution disk as a guide for creating your own extensions.

8.3.1 CodeWarrior Pro 4 (PowerPC)

All the files you will need to build the sample Extensions using CodeWarrior Pro 4 are provided in the “Making Extensions” folder. The CodeWarrior project settings are discussed below.

Open the project’s Settings window.
Click on the choice “Target Settings”. Use these values:

<table>
<thead>
<tr>
<th>Target Name</th>
<th>the name of your project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linker</td>
<td>MacOS PPC Linker</td>
</tr>
<tr>
<td>Pre-linker</td>
<td>none</td>
</tr>
<tr>
<td>Post-linker</td>
<td>none</td>
</tr>
<tr>
<td>Output Directory</td>
<td>usually {Project}:</td>
</tr>
</tbody>
</table>

Click on the choice “PPC Target”. Use these values:

<table>
<thead>
<tr>
<th>Project Type</th>
<th>Code Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>your extension’s file name</td>
</tr>
<tr>
<td>Sym Name</td>
<td>may be left blank</td>
</tr>
<tr>
<td>Resource Name</td>
<td>best set to the same name as the file name.</td>
</tr>
<tr>
<td>Creator</td>
<td>IPLB (case is important!)</td>
</tr>
<tr>
<td>Type</td>
<td>IPLX (case is important!)</td>
</tr>
<tr>
<td>Res Type</td>
<td>IPMX (case is important!)</td>
</tr>
<tr>
<td>Res ID</td>
<td>a unique number in the range described above</td>
</tr>
<tr>
<td>Display Dialog</td>
<td>Unchecked</td>
</tr>
<tr>
<td>Merge To File</td>
<td>Unchecked: Add your resource files to the project window instead.</td>
</tr>
<tr>
<td>Header Type</td>
<td>None (Native also works, but None is safer if you weak link with other libraries).</td>
</tr>
<tr>
<td>Resource Flags</td>
<td>Preload and Locked</td>
</tr>
</tbody>
</table>

Click on the choice “PPC Processor”. Fill in these values:

<table>
<thead>
<tr>
<th>Struct Alignment</th>
<th>PowerPC (IPLabExtensions.h specifies 68K alignment for any structs that require it).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make Strings Read Only</td>
<td>Checked</td>
</tr>
<tr>
<td>Store Static Data in TOC</td>
<td>Checked</td>
</tr>
<tr>
<td>Use FMADD &amp; FMSUB</td>
<td>Checked</td>
</tr>
<tr>
<td>other options</td>
<td>as you desire</td>
</tr>
</tbody>
</table>
Click on the choice “PPC Linker”. Fill in these values:

<table>
<thead>
<tr>
<th>Dead-strip Static Initialization Code</th>
<th>Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialization Entry Point</td>
<td>blank</td>
</tr>
<tr>
<td>Main Entry Point</td>
<td>main</td>
</tr>
<tr>
<td>Termination Entry Point</td>
<td>blank</td>
</tr>
<tr>
<td>other options</td>
<td>as you desire</td>
</tr>
</tbody>
</table>

Click on the choice “PPC PEF”. Fill in these values:

<table>
<thead>
<tr>
<th>Export Symbols</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code Sorting</td>
<td>None</td>
</tr>
<tr>
<td>version numbers</td>
<td>best kept all zeros</td>
</tr>
<tr>
<td>Fragment Name</td>
<td>blank</td>
</tr>
<tr>
<td>Library Folder ID</td>
<td>0</td>
</tr>
<tr>
<td>Share Data Section</td>
<td>Unchecked</td>
</tr>
<tr>
<td>Expand Uninitialized Data</td>
<td>Unchecked</td>
</tr>
<tr>
<td>Collapse unused TOC-reloads</td>
<td>Checked (but not required)</td>
</tr>
</tbody>
</table>

**Resources:**
If your extension uses other resources (than code), you need to add these to the project window. Some resources, like the IP25 and IPmn resources, are easier to describe in text format. Others, like DLOG and DITL resources are easier to work with in applications that produce .rsrc files.

1. Use ResEdit (or a similar application) to produce a resource file containing the resources. Name this file something like “MyExtension.rsrc”. (Appending “.rsrc” to such a file is a standard Mac naming convention.)

   - OR –

   Define your resources in text format using the Rez language. Name this file something like “MyExtension.r”. (Appending “.r” to such a file indicates it uses the Rez language).

2. Use CodeWarrior’s “Add Files…” option to add the .rsrc and/or .r file(s) to the project window.
8.3.1.1 Callbacks

The header file provided on the distribution disk defines macros for all of the callbacks in the form of UniversalProcPtrs. To make a callback you need only use the macro name, along with an extra parameter which is the extension record pointer. For example, to call UpdateImage with the parameter `&imgRef` you would do this:

```
CBUpdateImage (p, &imgRef);
```

p is a pointer to the extension record passed in by IPLab. The callback definitions are expanded to the proper form by the compiler.

8.4 Sample Extensions

Sample extensions are supplied to give you a jump-start in your own development. We recommend that you begin with one of these extensions and gradually modify it to perform your own tasks.

8.4.1 Sample Extension 01

This extension sets the pixels within the ROI bounds of the active image window (Byte type only) to a value that has been set in the Variables window (Var 0). If the window has multiple frames, it sets the value in each frame, as well.

To Run:

1. Use the New Image command to create a new byte image. 320 x 240 is fine. Specify more than one frame if you want.
2. Define an ROI within the image using one of the ROI tools.
3. Show the Variables window (Window menu) and set the first value (0) to a number below 255, say 100 (Click on the first value, or use IPLab’s Define ROI command to set the ROI to the first element. Then use the Set ROI Value command to change its value.)
4. Bring the image window to the front by clicking on it.
5. Select Sample 01 from the Ext. menu.

The extension will change the values within the ROI of each frame to the given value.

8.4.2 Sample Extension 02

This extension runs through the list of visible image windows and sets the ROI in each to a specific rectangle.

To Run:

1. Open a number of image windows, or create new ones.
2. Set the ROI in each to a different rectangle.
3. Select the extension from the Ext. menu.
4. The ROI in each window will be set to the same rectangle.
8.4.3 Sample Extension 03

This extension has two commands. Each creates a new window with three columns and 24 rows. The first column lists 24 angles between 0 and 360 in 15° increments, the second column holds the sines of those angles and the third holds the cosines of those angles. The first command displays this window in text view format, and the second displays the window as a graph.

To Run:

1. Select one of the two commands from the Ext. menu.
2. The new window will appear in either text view or as a graph.

8.4.4 Sample Extension 04

This extension is similar to 01, but gets the value from a dialog. It only operates on the current frame. For example’s sake, the number is derived from an operation on two numbers entered in the dialog.

To Run:

1. Create a new image window, and an ROI within it, as for Sample 01.
2. Select the extension from the Ext. menu.
3. A dialog will appear with two edit fields, and a pop-up menu between them. The pop-up specifies which arithmetic operation to perform on the two numbers. Enter two numbers, an operation, and remember the result. When you click OK, the pixels within the ROI bounds of the window will be set to that result.
4. To see the command execute from a script, create a new script window.
5. Select the extension from the Ext. menu and enter two new numbers.
6. After pressing OK, the name of the extension will appear in the script window.
7. Double-click that line to see the dialog, with the two numbers you entered. Change the numbers and press OK to have the new values replace the old.
8. Execute the script to perform the command with your two new numbers.

8.4.5 Sample Extension 05

Sample extension number 5 is included to show how you can create and access overlays. It also stores information in a resource between launches of IPLab. This extension adds an overlay to an image window, erases it, and draws a line of text into the overlay. The text gives the number of seconds since the last execution of the command.

To Run:

1. Use the New Image command to create a new image of any type. A size of 320 x 240 is fine.
2. Select Sample 05 from the Ext. menu.
   The process will draw “Seconds since last run” in yellow in the upper-left corner of the image, followed by the number of seconds in magenta.
3. Run the command a second time to see the time change.
4. Quit IPLab, then start it again and repeat steps 1 and 2 to see the new time.
8.4.6 Sample Extension 06

Sample extension number 6 shows how to use the fDataMinPtr and fDataMaxPtr fields.

To Run:

1. Show the Status palette (Window menu) if it is not already open.
2. Use the New Image command to create a new image of Byte type. 320 x 240 is fine.
3. Select Sample 06 from the Ext. menu. A dialog will display the current values of the min. and max. fields. If you press OK, the Min and Max fields in the Status palette will change to 123 and 231, respectively.
4. Use the Change Data Type command (Math menu) to change the window’s data type to Short Int. The Min and Max values will be reset to 0.
5. Select Sample 06 again. The dialog will show the initial values of the min. and max. fields. After pressing OK, the Min and Max fields in the Status palette will change to -12312 and 23123, respectively.
6. Repeat steps 4 and 5 for each additional (non-color) data type.
   The values will be:
   - Unsigned 16: 12312 23123
   - Long Int.: -123123 231231
   - Floating Pt.: -1.23e+02.31e+0

8.4.7 Sample Extension 07

Sample Extension 7 shows the use of a callback to get the fRefCon field of another extension (in this case, Sample 05). The result is that this extension can share data with another.

To Run:

1) Use the same window from Sample 05.
2) Select Sample 07 from the Ext. menu.
3) A message will be drawn into the overlay giving the last date and time at which Sample 05 was run (this, in fact, is the actual value stored by Sample 05).
8.4.8 Sample Extension 08

This extension is similar to 04, but it implements two commands. The first is the same as in Sample 04, but it also stores the last values used so that they can be used the next time the command is run. The second command does not have a dialog, but instead uses the values last set by the first command.

To Run:

1) Create a new image window, and an ROI within it, as for Sample 01.
2) Select the Sample 08 command Fill ROI… from the Ext. menu.
3) The same dialog used in Sample 04 will appear. Enter two numbers and an operation. When you click OK, the pixels within the ROI bounds of the window will be set to that result.
4) Select the other Sample 08 command Fill Non-ROI from the Ext. menu. The command will set the pixels outside the ROI to the same value used inside the ROI.
5) Select the Fill ROI… command again. The dialog will appear showing the settings that were entered previously. Change the values in the dialog, and press OK.
6) To see these commands execute from a script, create a new script window.
7) Select the Set Non-ROI Value command from the Ext. menu and the command name will appear in the script window. Then, select the Set ROI Value… command from the Ext. menu. The dialog will appear showing the settings that were used earlier.
8) Change the numbers and press OK. The command name will appear in the script window.
9) When you execute the script, the pixels outside the ROI will be set to the old value that was last used for pixels inside the ROI and the pixels inside the ROI will be set to the new value that you have specified.
## Appendix

### 9.1 Command Key Summary

<table>
<thead>
<tr>
<th>Key</th>
<th>Command</th>
<th>Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>Tile</td>
<td>Window</td>
</tr>
<tr>
<td>=</td>
<td>Set ROI Value</td>
<td>Edit</td>
</tr>
<tr>
<td>1</td>
<td>Measure ROI</td>
<td>Analyze</td>
</tr>
<tr>
<td>2</td>
<td>Segment at ROI</td>
<td>Analyze</td>
</tr>
<tr>
<td>A</td>
<td>Select All</td>
<td>Edit</td>
</tr>
<tr>
<td>C</td>
<td>Copy (ROI)</td>
<td>Edit</td>
</tr>
<tr>
<td>D</td>
<td>Duplicate Window</td>
<td>Window</td>
</tr>
<tr>
<td>E</td>
<td>Pseudocolor</td>
<td>Enhance</td>
</tr>
<tr>
<td>F</td>
<td>Select Frame</td>
<td>View</td>
</tr>
<tr>
<td>G</td>
<td>Run Script</td>
<td>Script</td>
</tr>
<tr>
<td>H</td>
<td>Home Image</td>
<td>View</td>
</tr>
<tr>
<td>K</td>
<td>Dispose Window</td>
<td>Window</td>
</tr>
<tr>
<td>L</td>
<td>Split Color Chan.</td>
<td>Math</td>
</tr>
<tr>
<td>M</td>
<td>Merge Color Chan.</td>
<td>Math</td>
</tr>
<tr>
<td>N</td>
<td>New Image</td>
<td>File</td>
</tr>
<tr>
<td>O</td>
<td>Open</td>
<td>File</td>
</tr>
<tr>
<td>P</td>
<td>Print</td>
<td>File</td>
</tr>
<tr>
<td>Q</td>
<td>Quit</td>
<td>File</td>
</tr>
<tr>
<td>R</td>
<td>Rename Window</td>
<td>Window</td>
</tr>
<tr>
<td>S</td>
<td>Save As</td>
<td>File</td>
</tr>
<tr>
<td>T</td>
<td>Convert to 8/24 Bit</td>
<td>Math</td>
</tr>
<tr>
<td>U</td>
<td>Show/Hide Status</td>
<td>Window</td>
</tr>
<tr>
<td>V</td>
<td>Paste</td>
<td>Edit</td>
</tr>
<tr>
<td>W</td>
<td>Close</td>
<td>File</td>
</tr>
<tr>
<td>X</td>
<td>Cut (ROI)</td>
<td>Edit</td>
</tr>
<tr>
<td>Y</td>
<td>Change Window</td>
<td>Window</td>
</tr>
<tr>
<td>Z</td>
<td>Undo(^a)</td>
<td>Edit</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Command</th>
<th>Menu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Change Window</td>
<td>Window</td>
</tr>
<tr>
<td>Delete</td>
<td>Clear (ROI)</td>
<td>Edit</td>
</tr>
<tr>
<td>W</td>
<td>Close</td>
<td>File</td>
</tr>
<tr>
<td>T</td>
<td>Convert to 8/24 Bit</td>
<td>Math</td>
</tr>
<tr>
<td>C</td>
<td>Copy (ROI)</td>
<td>Edit</td>
</tr>
<tr>
<td>X</td>
<td>Cut (ROI)</td>
<td>Edit</td>
</tr>
<tr>
<td>K</td>
<td>Dispose Window</td>
<td>Window</td>
</tr>
<tr>
<td>D</td>
<td>Duplicate Window</td>
<td>Window</td>
</tr>
<tr>
<td>H</td>
<td>Home Image</td>
<td>View</td>
</tr>
<tr>
<td>I</td>
<td>Measure ROI</td>
<td>Analyze</td>
</tr>
<tr>
<td>M</td>
<td>Merge Color Chan.</td>
<td>Analyze</td>
</tr>
<tr>
<td>N</td>
<td>New Image</td>
<td>File</td>
</tr>
<tr>
<td>O</td>
<td>Open</td>
<td>File</td>
</tr>
<tr>
<td>V</td>
<td>Paste</td>
<td>Edit</td>
</tr>
<tr>
<td>P</td>
<td>Print</td>
<td>File</td>
</tr>
<tr>
<td>E</td>
<td>Pseudocolor</td>
<td>Enhance</td>
</tr>
<tr>
<td>Q</td>
<td>Quit</td>
<td>File</td>
</tr>
<tr>
<td>R</td>
<td>Rename Window</td>
<td>Window</td>
</tr>
<tr>
<td>G</td>
<td>Run Script</td>
<td>File</td>
</tr>
<tr>
<td>S</td>
<td>Save</td>
<td>File</td>
</tr>
<tr>
<td>2</td>
<td>Segment at ROI</td>
<td>Analyze</td>
</tr>
<tr>
<td>A</td>
<td>Select All</td>
<td>Edit</td>
</tr>
<tr>
<td>F</td>
<td>Select Frame</td>
<td>View</td>
</tr>
<tr>
<td>=</td>
<td>Set ROI Value</td>
<td>Edit</td>
</tr>
<tr>
<td>U</td>
<td>Show/Hide Status</td>
<td>Window</td>
</tr>
<tr>
<td>L</td>
<td>Split Color Chan.</td>
<td>Analyze</td>
</tr>
<tr>
<td>;</td>
<td>Tile</td>
<td>Window</td>
</tr>
<tr>
<td>Z</td>
<td>Undo(^a)</td>
<td>Edit</td>
</tr>
</tbody>
</table>

\(^a\) Undo is not currently implemented.
# 9.2 Numeric Variables Reserved by *IPLab*

Some *IPLab* commands store their data in variables. You can use these variables, too; just remember that *IPLab* may overwrite any information you place in these variables. We do encourage you to reference these variables when writing your own scripts. For example, it's very useful to employ variable #255, the last exposure time used, within the **Full Acquire** command.

<table>
<thead>
<tr>
<th>Variable #</th>
<th>Information Stored in This Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>ROI left, recorded by <strong>Extract: Image Sizes</strong> (<a href="#">Analyze menu</a>)</td>
</tr>
<tr>
<td>201</td>
<td>ROI top</td>
</tr>
<tr>
<td>202</td>
<td>ROI right</td>
</tr>
<tr>
<td>203</td>
<td>ROI bottom</td>
</tr>
<tr>
<td>204</td>
<td>ROI width</td>
</tr>
<tr>
<td>205</td>
<td>ROI height</td>
</tr>
<tr>
<td>206</td>
<td>Image width</td>
</tr>
<tr>
<td>207</td>
<td>Image height</td>
</tr>
<tr>
<td>208</td>
<td>Current Z (Current frame)</td>
</tr>
<tr>
<td>209</td>
<td>Z depth (Total frames)</td>
</tr>
<tr>
<td>210-211</td>
<td></td>
</tr>
<tr>
<td>212</td>
<td>Data Type</td>
</tr>
<tr>
<td>213</td>
<td>XY Unit Factor</td>
</tr>
<tr>
<td>214-231</td>
<td></td>
</tr>
<tr>
<td>232</td>
<td>Current index used by the <strong>Iterator</strong> tab in <strong>StageScan Move/Iterator</strong> (<a href="#">Control menu</a>)</td>
</tr>
<tr>
<td>233</td>
<td>Number of positions to be moved by the <strong>Iterator</strong> tab in <strong>StageScan Move/Iterator</strong> (<a href="#">Control menu</a>)</td>
</tr>
<tr>
<td>234</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td>Current row position within the array, as set by the <strong>Iterator</strong> tab in <strong>StageScan Move/Iterator</strong>. Current row position within the super-array, as set by the <strong>Move</strong> tab.</td>
</tr>
<tr>
<td>236</td>
<td>Current column position within the array, as set by the <strong>Iterator</strong> tab, Current column position within the super-array, as set by the <strong>Move</strong> tab.</td>
</tr>
<tr>
<td>237</td>
<td>Current row position within the super-array, as set by the <strong>Iterator</strong> tab, Current row position within the array, as set by the <strong>Move</strong> tab.</td>
</tr>
<tr>
<td>238</td>
<td>Current column position within the super-array, as set by the <strong>Iterator</strong> tab, Current column position within the array, as set by the <strong>Move</strong> tab.</td>
</tr>
<tr>
<td>239 - 251</td>
<td></td>
</tr>
<tr>
<td>252</td>
<td>Exposure time for Red channel for <strong>Acquire 3 Channel Color</strong> (<a href="#">Camera menu</a>)</td>
</tr>
<tr>
<td>253</td>
<td>Exposure time for Green channel for <strong>Acquire 3 Channel Color</strong></td>
</tr>
<tr>
<td>254</td>
<td>Exposure time for Blue channel for <strong>Acquire 3 Channel Color</strong></td>
</tr>
<tr>
<td>255</td>
<td>Exposure time for main <strong>Acquire</strong> command</td>
</tr>
</tbody>
</table>
# 9.3 The *IPLab* Data File Format

## 9.3.1 Data File Readers

This file format information is supplied for programmers who wish to read *IPLab* data files with other software. Scanalytics reserves the right to change this file format as needed.

*IPLab* data files have type IPLI and the *IPLab* creator designation is IPLB (as seen by using Get Info in ResEdit). Each IPLI data file has a header of 2120 bytes, which is described by the following table.

<table>
<thead>
<tr>
<th>Start Byte</th>
<th>Field Description</th>
<th>Field Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Version</td>
<td>4-Byte ASCII field. Version 3.2 value = ‘3.2’</td>
</tr>
<tr>
<td>4</td>
<td><em>IPLab</em> File Format</td>
<td>Unsigned Byte. 0 for data files.</td>
</tr>
<tr>
<td>5</td>
<td>Data Type</td>
<td>Unsigned Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0=Unsigned Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1=Short (16-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2=Long (32-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3=Floating Point (32-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4=Color16 (16-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5=Color 24 (32-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6=Unsigned Short (16-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7=MDSqrt (16-bits/pixel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8=Color 48 (48-bits/pixel)</td>
</tr>
<tr>
<td>6</td>
<td>Data Width</td>
<td>Long Int (4 Bytes)</td>
</tr>
<tr>
<td>10</td>
<td>Data Height</td>
<td>Long Int (4 Bytes)</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>6 Bytes</td>
</tr>
<tr>
<td>20</td>
<td>nFrames</td>
<td>Short Integer--Number of frames</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>50 Bytes</td>
</tr>
<tr>
<td>72</td>
<td>Start 1D Data (if 1D)</td>
<td>No. of Samples*No. Bytes/sample</td>
</tr>
<tr>
<td>72</td>
<td>CLUT (if 2D)</td>
<td>2048-Byte array</td>
</tr>
<tr>
<td>2120</td>
<td>Start Image Data (if 2D)</td>
<td>Width * Height <em>(No. Bytes/pixel)</em> nFrames</td>
</tr>
<tr>
<td>2120 + image size</td>
<td>Overlay Size in Bytes</td>
<td>Long Int (4 Bytes)</td>
</tr>
<tr>
<td>varies</td>
<td>Data Trailer</td>
<td>Overlay, Registration marks, the ROI, etc.</td>
</tr>
</tbody>
</table>
The data directly follows the header in raw binary format and is organized in a row-ordered fashion with no extra characters between rows:

\[(0,0), (1,0), \ldots, (\text{Width}-1,0), (0,1), (1,1), \ldots, (\text{Width}-1,1), \ldots, (0,\text{Height}-1), (0,\text{Height}-1), \ldots, (\text{Width}-1,\text{Height}-1)\]

If there is more than one frame \((\text{nFrames}>1, \text{i.e.} \text{an image sequence})\), they are stored immediately after the first frame in the same fashion with no inter-frame gaps.

Following the data is a “trailer” which contains various other information which generally cannot be interpreted by other programs. Your image reader program should ignore all information after the data.

### 9.3.2 Data File Writers

When IPLab was first introduced in 1989, Scanalytics Inc. (formerly Signal Analytics Corporation) could not find a suitable existing file format that gave us the flexibility that we needed to store image data with 8, 16 or 32 bit precision, color data and the option to store image sequences. Hence the IPLab file format was born (the Macintosh code for this file type is IPLI).

Many users have expressed an interest in being able to write image data in the native IPLab for Macintosh file format from other software packages. We recommend that you first consider creating an image write routine that uses one of the more widely familiar file formats, such as TIFF or raw binary data. Many software packages read and write TIFF files, and the specification allows for the differences between various kinds of computers (Macintosh and IBM PC compatibles have different byte ordering for data values).

However, it is true that to create a proper TIFF writer requires substantial programming effort. The next best answer, and an easier thing to do, is to write raw binary files. Since IPLab can read raw binary files as “foreign files”, and it can even extract the image size information from a header, this is an attractive technique for writing files. If you choose this method, make the files with a simple header before the data like this:

- **width**: (16 bit integer value)
- **height**: (16 bit integer value)
- **nFrames**: (16 bit integer value)
- **data...**: (pixel 1, pixel 2, ... row-ordered)

In IPLab version 3.2 you can define foreign file types that easily reads such files. Although the simple header described above contains almost all of the information IPLab needs to open this data file, one piece is missing: the data type (8 bit unsigned, 16 bit signed, 16 bit unsigned, etc.). This means that in IPLab you must create a separate foreign file type for each data type, and the user must be told separately what type of data this is so s/he can use the proper foreign file type when opening the file. You can image that this simple header could be extended to include a field which gives a code for the file type also. At this time IPLab does not decode data type information in a foreign file header, primarily because there is no universally accepted code for data type. Nevertheless, you can expect that such a simple file type will have more of a future in upcoming releases of IPLab.

However, if you are still interested in creating software which will write an IPLab native file, we include here the specifications you will need to write such a file.

The IPLab data file format is quite complicated and has changed many times since IPLab was first introduced. At the beginning it consisted of a fixed length header followed by a color table then the image data. Since that time, we have had to add so much more information to the file that the header could not hold it all. Instead of expanding the header size, we chose to append information behind the data in a “trailer.” Some of this information is specialized and coded and may be difficult to generate in a simple software package. Therefore we insist that you ignore the trailer information and concentrate on the basic header and data information. For that reason, we provide here the file format as it appeared in an earlier version of IPLab for Macintosh. As each new version of IPLab for Macintosh has been introduced, we have maintained backwards compatibility with IPLI files. So if you create image files using this information, users of IPLab for Macintosh version 2.5 and later will be able to read your files as if they were made by IPLab itself.
**Resource Information**

*type:* IPLI  
*creator:* IPLB

**Header Information**

<table>
<thead>
<tr>
<th>Field</th>
<th>C Type</th>
<th>{Bytes this field/total}</th>
<th>Required values if any</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>OSType</td>
<td>{4/4}</td>
<td>‘2.5a’</td>
</tr>
<tr>
<td>format</td>
<td>Byte</td>
<td>{1/5}</td>
<td>0</td>
</tr>
<tr>
<td>dType</td>
<td>Byte</td>
<td>{1/6}</td>
<td>0=Byte, 1=Signed Short, 2=Long, 3=Float, 5=Color 24, 6=Unsigned Short, 8=Color 48</td>
</tr>
<tr>
<td>nWidth</td>
<td>Long</td>
<td>{4/10}</td>
<td></td>
</tr>
<tr>
<td>nHeight</td>
<td>Long</td>
<td>{4/14}</td>
<td></td>
</tr>
<tr>
<td>fileClutID</td>
<td>Byte</td>
<td>{1/15}</td>
<td>0=standard, 1=reverse monochrome</td>
</tr>
<tr>
<td>overlayInFile</td>
<td>Boolean</td>
<td>{1/16}</td>
<td>0</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/17}</td>
<td>0</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/18}</td>
<td>0</td>
</tr>
<tr>
<td>viewMode</td>
<td>Byte</td>
<td>{1/19}</td>
<td>0=image, 1=text</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/20}</td>
<td>0</td>
</tr>
<tr>
<td>nFrames</td>
<td>Short</td>
<td>{2/22}</td>
<td></td>
</tr>
<tr>
<td>delta</td>
<td>Double</td>
<td>{8/30}</td>
<td>1.0</td>
</tr>
<tr>
<td>reserved</td>
<td>Short</td>
<td>{2/32}</td>
<td>0</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/33}</td>
<td>0</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/34}</td>
<td>0</td>
</tr>
<tr>
<td>units</td>
<td>Pascal String</td>
<td>{10/44}</td>
<td>‘Pixels’</td>
</tr>
<tr>
<td>reserved</td>
<td>Byte</td>
<td>{1/45}</td>
<td>0</td>
</tr>
<tr>
<td>normType</td>
<td>Byte</td>
<td>{1/46}</td>
<td>1=linear</td>
</tr>
<tr>
<td>normSource</td>
<td>Byte</td>
<td>{1/47}</td>
<td>1=frame min/max</td>
</tr>
<tr>
<td>numRegMarks</td>
<td>Byte</td>
<td>{1/48}</td>
<td>0</td>
</tr>
<tr>
<td>normMin</td>
<td>Double</td>
<td>{8/56}</td>
<td>data minimum value or 0</td>
</tr>
<tr>
<td>reserved</td>
<td>LongInt</td>
<td>{4/60}</td>
<td>0</td>
</tr>
<tr>
<td>normMax</td>
<td>Double</td>
<td>{8/68}</td>
<td>data maximum value or 4095 for 12 bit data</td>
</tr>
<tr>
<td>reserved</td>
<td>LongInt</td>
<td>{4/72}</td>
<td>0</td>
</tr>
<tr>
<td>Color Table</td>
<td>CSpecArray</td>
<td>(2048/2120)</td>
<td>any legal color table</td>
</tr>
</tbody>
</table>

The fields that are of type Double assume IEEE standard floating point double precision values.

The data directly follows the header in raw binary format and is organized in a row-ordered fashion with no extra bytes between rows:

\[(0,0), (1,0),..., (\text{Width}-1,0), (0,1), (1,1),..., (\text{Width}-1,1),...(0,\text{Height}-1), (0,\text{Height}-1),..., (\text{Width}-1,\text{Height}-1)\]

If there is more than one frame (nFrames>1, *i.e.* an image sequence), they are stored immediately after the first frame in the same fashion with no inter-frame gaps.

If the data is Color 24, then it must be stored as “chunky” 32 bit data. That is, each pixel contains 4 bytes formatted as: [blank byte, red byte, green byte, blue byte].

If the data is Color 48, then it must be stored as “chunky” 48 bit data. That is, each pixel contains 3 words (6 bytes) formatted as: [red word, green word, blue word].
9.4 Script Converter

The *IPLab* Script Converter is a utility application which converts scripts from version 2.3 (and later) of *IPLab* to version 3.6. This application converts any script commands that have changed and lists any items that you may need to examine before using the script.

9.4.1 The Script Converter Window

![The Script Converter Window](image)

The Script Converter runs from a single window. The window is divided into three parts:

**Status:** The top pane shows what the Script Converter is currently doing (“Status”) and on which file it is operating (“Converting”). If the converter is idle, the script name is replaced with a button that allows you to select a script to convert.
When you click on Choose A Script..., the standard file opening dialog is presented, with an additional check box:

Choose a Script to Convert

If you check Convert All Scripts In This Folder then all scripts which are contained within the folder named at the top of the window are converted, one by one, without any further work on your part.

* Remember, the folder referred to by the check box is the one named in the pop-up at the top of the window, not any folder you might have selected in the list.

Messages: The large box contains a scrolling list of messages that appear during the conversion process. The messages show the name of the file being converted, the name of the new file, and any instructions or notes about the new script. The messages can be saved to a text file with the Save As... button. The resulting report file can be opened by SimpleText (or TeachText), or any word processor or text editor. Such applications may also be used to print the report. The Clear button erases the current contents of the list. Note that the contents of the list are also cleared each time the converter is closed.

Options: At the bottom of the window are three pop-up menus which are used to set how the Script Converter works. The first (Convert to) specifies the version of IPLab with which the new scripts will run. The second pop-up (Output Rule) allows you to choose the rule for naming the new script file.

* These options are “remembered” by the Script Converter, and are used each time it is opened.

Output Rule: There are three ways in which the Script Converter can produce new script files from old:

Append version # to old script: This choice renames the existing script by appending the version of IPLab that created it (for example, “(3.0)”) to the end of its name. The new script is then named with the original script’s name.

Append version # to new script: This choice renames the new script by appending the version of IPLab listed in the “Convert to” option [for example “(3.5)” to the end of the original name]. The old script retains its original name.

Overwrite old script with new: This option replaces the existing script with the new one. The file’s original contents are lost.

* If you are going to convert a folder full of scripts, it is best to duplicate the folder, choose Overwrite old script with new, and then use the Convert all scripts in this folder feature. This makes it easier to separate out the old and new scripts after the conversion is done.

Menus: The Clear item in the Edit menu is identical to the Clear button in the message box. In the File menu are items which match the Choose A Script, and Save As buttons in the main window. Clicking the close box in the window is the same as choosing Quit from the File menu. The About command in the Apple menu shows the current version of the application.
9.4.1.1 Using the Script Converter

First, review the options in the pop-ups at the bottom of the window and change any that are incorrect. If you used a frame grabber command in one or more of your scripts (any that appear in IPLab’s Control menu) you should also select the frame grabber which you used when you produced your script(s).

Now click on the Choose A Script… button and select a script to convert. The conversion begins as soon as the file selection dialog is closed. The messages window lists which file is being converted, what the old and new file names will be, and any messages specific to the commands in the script.

That’s it! Once the conversion is complete you may convert more or Quit the converter.

* Remember to save any messages before quitting the program. They are not saved between launches.

9.4.2 Miscellaneous

9.4.2.1 Converting Extensions

The following extensions are converted by the Script Converter to work with IPLab version 3.6 and later:

3D: these commands are updated to work with the version of 3D that ships with IPLab 3.6.

Timer: is converted to the built-in Pause command.

U-Enter Variables: is converted to the built-in Enter Variables command.

No other extensions or camera control are converted or updated. The Script Converter simply copies their parameters directly to the new script, without making any alterations to them.

IMPORTANT: Although the script is converted, this does not guarantee that the extension or camera control itself is in fact compatible with IPLab for Macintosh version 3.6 or later. If you must update your extensions or camera controls to make them compatible with version 3.6 or later, then their dialog parameters may have changed. This means that the corresponding command lines in the converted script would send wrong parameters to the extensions or camera control commands. Therefore if you update your extensions or camera controls, you must remove the corresponding commands from your scripts and re-insert them after you do the update.

9.4.2.2 Conversion of Loop, If-Then and Jump from 2.3 Scripts

In version 2.3, the script commands Loop, If-Then, and Jump used line counts to specify where execution would continue. Versions 2.4 and later use labels. The Script Converter converts the three branching commands to use labels, and inserts the necessary labels. The new labels are named “Label A”, “Label B”, etc. If you have more than 26 labels, the names become “Label AA”, Label AB”, and on up to ZZ.

9.4.2.3 Converting Frame Grabber Commands for Pre-3.0 Scripts

If you are converting from a script that was written for IPLab for Macintosh version 2.5.7 or earlier, any frame grabber commands will be lost. You will have to enter them into the new script. Frame grabber commands in scripts from version 3.0 and later are converted in the same way extensions are.
10  Index

# / V button, 119, 121
+- symbol (in scripts), 100, 139
3D, 268
   3D reconstruction, 268
   3D scripts, 286
3D Equalize, 269
3D Filter, 271
3D Merge Sequences. See Merge Color Channels
3D Mosaic, 276
3D Oblique View, 278
3D Projector, 283
3D ROI Sum Plot, 275
3D Stacked View, 279
3D Tracer, 280
   Examples, 281
3-D View, 205, 206
About IPLab, 154, 333
Action Variables, 292
Active window. See Front Window
Adaptive filtering, 273
Add two images, 258
Adjust Contrast, 38, 41, 209
Affine Color Transform, 251, 266
Align. See Register
Analyzing Regions, 152
Angles, 90
Animate, 200
Ants, Marching. See ROI
AOI. See ROI
Apple events, 153
AppleScript, 153
Apply Color Table, 216
Area fractions, 89
Area of holes in particles, 88
Area of Interest. See ROI
Arithmetic, 258
Arrow tool, 35, 36
Arrows
   Arrow keys, 121
Asking the user a question, 107
Assign F(unction) Keys, 26, 196
   usage, 103, 113
Background subtraction, 96, 212
Background task, 153
Barrel Shift, 246, 255
Batch file processing, 114, 123, 158, 169
Binarize, 233, 256
Binary Morphology, 64, 86, 233
   Close, 66
   Complement, 67
Dilate, 67
Erode, 66
Hit & Miss, 66
Mask, 66
Open, 66
Skeletonize, 67
   Structuring element, 66
Blend color channels, 265
Boundary detection, 67
Brightness and contrast, 43
Byte data, 129
Calibrate Intensities, 247
Calibrating distance, 31, 76. See also Define XY Units
Calibration bar, 77
Cancel All, 294
Cell changes over time, 89
Change Color Type, 133, 182, 184, 252
Change Data Type, 252
Change Drawings, 193
Change Segment Color, 234
Change Window, 299
   in a script, 99
Circles, 35
Circular ROI, 32
Circularity, 239, 240
Classification, 238
   Particles, 78
Cleaning segments, 63, 64, 86
Clear, 185
   In Scripts, 98
Clear All Drawings, 193
Clear Selected Drawings. See Clear All Drawings
Clipboard, 134, 182, 183
Close Image, 160
Close Window, 296
Closing images, 42
CLUT, 44, 252
   Changing, 160, 312
   Changing, 14, 24, 44
   Conventional grayscale, 255
   Definition, 129
   During Paste, 184
   Extracting, 249
   Extracting the color table, 44
   Make your own, 45
   Pseudocoloring, 130
   Saving, 128
Co-localization, 265
Color
   Analyzing, 99
Choose color for segments and drawing, 36
Color 24 data, 46, 129
Color 48 data, 129
Color bar in the Tools window, 58
Color channels, 263, 266
Color coordinates, 130
Color look up table. See CLUT
Color transform, 266
Colorize, 214
Data Types, 129
Image enhancement, 46
merging and overlaying, 56
Segmentation, 233
Color Balance, 209
Color Channels, 264, 266
Color to Grayscale, 216
Column ordered measurements, 86, 238
Column ordered results, 89
Command. See names of individual commands
Command key, 121
   key combination summary, 341
Complementary segments, 67
Complex, 251, 259
Computer compatibility. See System requirements
Connected sets of pixels, 59
Continue, 294
Contours, 244
Contrast enhancement, 38, 40
   16-Bit grayscale, 42
   8-Bit grayscale, 38
   Color images, 46
   Contrast function, 38
   Gamma, 40
   Mid point, 40
Contrast Equalize, 211
Control key, 121
Control menu, 288
Convert color channels, 251, 266
Convert old scripts, 346
Convert to 8/24 Bit, 251
Coordinates, 10, 28, 31, 246, 249, 311
   Extracting, 250
Copy, 181, 182
   Copy and paste QGraphs, 84
   Copy and paste segments, 61
   Perspective View, 206
   QGraph, 205
Correlation filter, 109
Cosine transform, 245
Count segments, 242
Crop Image, 186, 298
Cut, 181
Cut, copy, paste on segments and Drawings, 35, 36
   DAPI images, 54
Data elements, 222. See Pixels
Data type, 28, 31, 129, 134, 155, 184, 251
   Changing, 252
Export, 166
Text files, 124
Export View
  Perspectives, 206
  QGraph, 205
Extensions, 303
  Architecture, 303
  Assigning to function key, 196
Callbacks, 312
  AddOverlay, 318
  AlterImageDimensions, 322
  CopyImageRef, 313
  DisposeImage, 316
  EndOverlayDraw, 318
  GetEmptyImageRef, 313
  GetExtRefCon, 331
  GetFrontImageRef, 313
  GetImageFSSpec, 321
  GetImageUnits, 331
  ImgToWinCoord, 322
  NewImage, 314
  NewImage2, 315
  NextImage, 316
  OpenImage, 319
  OperatingMode, 331
  ReleaseImageRef, 313
  SaveImage, 320
  SelectFrame, 322
  SetImageLabel, 317
  SetOverlayColor, 319
  ShowHideDisplay, 317
  ShowImage, 315
  ShowPalettes, 332
  StartOverlayDraw, 318
  UpdateImage, 316
  UpdateStatusWindow, 332
  UseAcquireData, 325
  UseImageMag, 324
  UseImageOffset, 323
  UseImageView, 324
  UseNormData, 326
  UseQGraphData, 327
  WinToImgCoord, 323
Development Environments, 334
Execution Sequence, 333
global variables, 333
  Image References, 309
  Loading, 122
  memory allocation, 334
  Parameter Block, 306
Resources, 303
  saving parameters, 333
Startup, 333
  using fRefCon, 333
Window Library, 332
Extract Image Sizes, 250
Extract ROI to 1D, 249
Extract submenu, 93, 249
  Linear ROI, 190
False color, 130
FFT Filter, 51, 219
File format, 343
File types, 127
Files. See also Windows
  Automatically numbering, 114
  File list, 150, 157, 158, 162, 165, 169, 171
  Path information, 169
FITS, 177
  Foreign or Raw, 173
  Formats, 127
  Indexed, 172
  Opening, 156
  Reading Foreign, 160
  Reading with File List, 157
  Saving, 160, 161
  Script, 289, 290
  Text, 123
Fill. See Set Value
  Fill holes in segments, 66
  Filled objects, 35
Film scanning
  Density calibration for, 94
Filter
  Edge, 218
  Editing or creating your own, 51
  FFT, 51
  kernels, 51
  Linear, 217, 222, 254
  Median, 218
  Morphological, 221
  Filter out small segments, 66
  Filtering, 50
  Filtering color images, 51
  Filtering Segments, 64
  Filters
    Adaptive filtering, 273
  First image. See Front window
FITC, 56
FITS, 128
  FITS Header, 177
FKey Toolbar, 26
FKeys
  Show/Hide FKeys, 295
Flat Field, 212
Flat field correction, 96
Flip, 54, 224
Floating data, 129
Fluorescence ratio imaging, 213
Font, 35
Foreign files, 128, 173
Foreign Formats, 173
Format of data files, 343
Fourier analysis
  real time, 117
Fourier transform, 245
Frames, 10
Frames to Sequence, 299
Freehand drawing, 36
Freehand ROI tool, 15
Freehand shape, 35
Front window, 9
Function keys. See Assign F(unction) Keys
  Toolbar, 26
Gamma, 40
Gamma function, 40
Gel analysis, 92
Geometric operations, 54
Geometric Rotate, 227
Geometric Scale, 227
Geometric transform, 224, 227
Geometric transformations, 225
Get information from the user
  Query command, 107
Grain boundaries, 234
Grayscale erosion, 96
Grayscale Morphology
    Dilate, 69
    Erode, 68
Hand tool. See Scroll tool
Hardware control, 288
Heading, row/column, 195
Hide Display, 184
Hide Drawing, 199
Hide FKeys, 295
Hide Image, 198
Hide Labels, 200
Hide ROI, 199
Hide Segment, 199
Hide Status, 295
Hide Tools, 295
Hide Variables, 296
High Dynamic Range Data, 207
Histogram, 90, 243
  joining multiple, 228
  real time, 117
  When you really want a slice of the pixels values, 92
Holes
  Area, 88
  Include in measurements, 88
  Measuring interior holes, 78
Home command, 60
Home image, 34, 198
HSV, 52
HSV, Hue-Saturation-Value, 131
Image
  Coordinates. See Coordinates
  Definition, 9
Image Arithmetic, 69, 258, 260, 261
  difference between two measurements, 88
Image info
  Define Image Info, 177
  Descriptor, 176. See Short Description
  Edit Image Info, 176
  Notes, 176
  Short Description, 177
Image Ratios, 212
Image view, 125
  Dithering, 201
Image window. See Windows
Import
  Text files, 123
  Including holes, 78
Indexed color, 129
Indexed Files, 114, 162, 169
  Changing the base name, 116
  Opening, 157
Installation, 2
  System requirements. See System requirements
Intensity, 78
Intensity along a line
  real time slice, 117
Interactive Operations, 121
Interleave Zeros, 254
Interpolate, 254
Invert, 213
IPL file format, 127
IPLab file format, 127
IPLab Mac file format, 128
Join images
  See Mosaic, 54
Key combination summary, 341
Label, row/column, 195
Labels, 197, 200, 241
Laser optics, 56
Length, 90
Limits on the size of particles to measure, 78
Line ROI tool, 15
Line tool, 190
Linear Filter, 50, 217, 222
  user defined kernels, 222
Linear ROI to 1D command, 93
Lines, 35
Long integer data, 129
Loop count
  Keeping track, 107
  Using it in a script, 108
Looping based on variables, 107
Magnify tool, 14, 24, 34, 60, 63
Manual measuring, 90
Manual selection tools, 15
Marching Ants. See ROI
Marquee. See ROI
Masks, 271
Matched Filter, 109
Max or min. between two images, 258
Max. number of segments, 86
Mean intensity, 78
Measure
  ROI, 241
  Segments, 242
  Segments inside the ROI, 81
Measure Angles, 14
Measure Lengths, 14
Measure ROI, 241
Measure Segments, 80, 152, 242
Measurement Options, 78, 241, 242
Measurement results
  Bring to front, 80
  Saving, 80
Measurements, 152, 237
  3D, 280
Measurements you can perform, 78
Measuring changes over time, 89
Measuring individual segments, 112
Measuring particles, 76
Measuring particles with holes, 85
Measuring single particles, 82
Median Filter, 50, 218
Memory, 3. See System requirements
Menu bar, 6
Merge Color Channel, 173
Merge Color Channels, 53, 56, 132, 264
Merge Color Coordinates. See Merge Color Channels
Merge Sequences. See Merge Color Channels
Microscope control, 288
Microscope objectives
  Storing a list of different ones, 76
Mid point, 40
Millions of colors, 46
Min/Max image values, 31
Minify. See Magnify tool
Minimum diameter, 240
Modify ROI, 191
Modify Segments, 86, 233
Moments, 78
Monochrome CLUT, 155
Monochrome image, 249, 255
Morphology, 97, 221
  Binary, 233
  Grayscale, 68
Mosaic, 54, 228
  3D Mosaic, 276
Move Window, 298
Movie, 200
Movies
  Exporting, 167
Moving draw objects, 35
Moving segments to another image, 70
Multi-line ROI tool. See Line ROI tool
Multiple regions, 194
Multiply two images, 258
Multi-spectral image data, 56
New Image, 155
New Script, 136, 289
Noise, 260
Noise reduction, 50
Noise removal by Median Filter, 218
Non-destructive overlay, 59
Nonlinearities in image capture and display, 40, 94
Normalization, 42, 207, 251, 310, 311
  Saving parameters, 128
Numbered files, 114
Oblique View. See 3D Oblique View
Open, 30, 150, 156, 169
  While Scripting, 99
Open As, 159
Open Script, 96, 289
Optimal lookup table, 41
Options, 125
Outlining particles, 79
Oval ROI tool, 15, 32
Oval Segment tool, 22
Overlay, 13, 35
  Change Color, 234
  Dispose, 248
  Show/Hide, 199
  Stamp on Image, 195
Page Setup, 178
Paintbrush tool, 35, 59
Parameters, 151
Particle Classification, 78, 238
Particle size, 240
Paste, 183
  Moving paste regions, 121
  Undo, 122
Paste Buffer, 134, 185
  and perspectives, 206
  and QGraph, 205
Pasting copied segment layer, 36, 61
Pattern matching, 109
Pause. See Delay
Persistence, 125
Perspective View, 205, 206
  export view, 166
  Options, 204
Photo, 200
PICT, 128
Piecewise linear ROI, 93
Pixel Arithmetic, 257
Pixels, 10, 184
Placing multiple segments, 60
Plot
  For reports, 83
  Plot view, 125
  Printing, 179
  Row/Column Average, 244
Point and Measure, 112
Polygon ROI tool, 15
Polygons, 35
Position, 78
Preferences, 125, 197
Preview while opening, 30
Print, 179
  Color Separations, 133
  printing plots, 179
Printing, 178
Process many images, 114
Project 3D. See 3D Projector
PSF file format, 127
QGraph, 83, 181, 205
  export view, 166
QGraph View
  Options, 202
Quantify Segments, 89, 242
QuickTime, 30
QuickTime Movies, 167
Quit, 29, 36, 57, 180
Rank filter
  Minimum, 68
Ratios, 212
Raw data, 128, 173
Real time image analysis, 117
Record Timer. See Delay
Rect Segment tool, 21
Rectangle ROI tool, 15, 60
Rectangles, 35
Region analysis, 194
Region of interest, 32, 40, 42. See ROI
Regions
  Transferring segments, 14
Register, 55, 225
Register for mosaic, 54
Register images, 25
Registration mark, 25
  Deleting, 122
Rejecting small particles, 240
Remote sensing, 56
Rename Window, 297
Requirements for installation. See System requirements
Re-size, 54
Resize the ROI, 33
Reverse Sequence, 193
Revert To Saved, 35, 164, 181
RGB, 131
ROI, 11, 31, 32, 152
  Boundaries, 12
  Defining, 15, 190
  Deleting, 122
  Dragging, 33, 61
  Extract data values, 249
  Extract Images Sizes, 250
  In Text windows, 83
  Measure, 241
  Moving, 13, 121
Multi-line, 249
  Piecewise linear, 93
Rectangle tool, 32
Resize, 33
Resize, 121
Selecting a column, 83
Setting color, 187
Setting value, 187
Show/Hide, 199
ROI boundary, 93
ROI Sum Plot. See 3D ROI Sum Plot
ROI tool
  Freehand, 32
ROI tools, 32. See ROI: Defining
ROI used to define Segment Copy, 36, 61
Rotate, 54, 225, 227
Rotate & Scale, 54, 227
Rotate CLUT, 14, 24, 44
Row/Col Plot, 244
  real time, 117
Row/Column Plots, 91
R,G,B,, 133
Run Script, 196, 290
  Using it in a script, 111
Saturated frame, 43
Save, 160
Save As, 160, 161, 169
  Using indexed files, 115
Saving measurements, 80
Scale (Geometric), 225, 227
Scale image size, 54
Script, 96, 135
  +/- symbols, 100
  3D scripts, 286
Alert button, 101
Allowing dialog interaction, 100
AppleScript, 153
Batch file processing, 123
Cancel All, 294
Closing one, 98
Continue, 294
Creating, 136
Drawing while running, 101
Editing, 98
End line, 97
Error handling, 150
File format, 128
Getting information at run time, 105
Highlighting commands in one, 98
IF command, 106
Interaction while running, 100
Interactive commands, 139
Label, 97, 105
Loop, 98
Making one, 97
Nested, 149, 290
New, 289
Open, 289
Place an ROI while running, 101
Query, 107
Query the user, 105
Quit- scripting the Quit command, 180
Record ON/OFF switch, 97
Run, 290
Run Script, 149
Run, Continue, Stop, 148
Running one from another, 149, 290
Stopping one while running, 98
Update/convert old scripts, 346
Using with F keys, 104
Using without opening through F keys, 113
Script Commands window, 97
Script Converter, 346
Scroll, 14, 34, 63, 200
Scroll image, 34
Scroll tool, 24
Segment at ROI, 60, 233
Segmentation, 229, 242, 243. See also Segments
   Binarize, 256
   Color Cube tab, 232
   Histogram tab, 231
   Manual segmentation tools, 20
   Manual tracing, 58
   Segment tools, 35
   Simple tab, 230
Segments, 58, 152, 233, 242. See also Segmentation
   Boundary only, 67
   Copy and paste, 61
   Creating by the Segmentation command, 62
   Deleting, 122
   Making. See Segmentation
   Measure, 242
   Segment layer, 59
   Segment layer, 13, 35
   Touching, 63
Select All, 188
Select All Drawings. See Select Next Drawing
Select File List, 171
Select Frame, 198
Select Next Drawing, 193
Select ROI. See ROI: Defining
Selecting draw objects, 35, 36
Selection. See ROI
Sequence, 31, 192, 198
Creating, 299
   Discard every other frame, 253
   Splitting, 300
Sequence to Frames, 300
Sequences, 10
   Navigating, 121
   Selecting frames, 9
Set Indexed Name, 157, 172
Set Measurements, 237, 242, 247
Set Pattern, 260
Set ROI Color, 187
Set ROI Value. See Set Value
Set Row/Column Label, 195
Set Value, 187
Set Variable, 291
Set XY Units, 77, 235
Setting units based on microscope objective, 106
Setting up measurements, 78
Settings, 125, 151
Shape, 78
Sharpen filter, 50, 51
Shift, 225
Short Int data type, 42
Short integer data, 129
Show Display, 184
Show Drawing, 199
Show FKeys, 295
Show Image, 198
Show Labels, 200
Show ROI, 199
Show Segment, 199
Show Status, 295
Show Tools, 295
Show Variables, 296
Signed Data Type, 251
Single Measure, 238
   Column ordered, 86
Single particle measuring, 82
Size distribution, 82
Size of drawing tools, 35
Sizes, 250
Skeletonize Segments, 67
Slice, 92, 93, 249
   real time analysis, 117
Slice (Extract linear ROI), 190, 249
Sort Data, 254
Split Color Channels, 52, 132, 152, 173, 251, 263
Split Color Coordinates. See Split Color Channels
Square ROI, 32
Square root, 291
Squares, 35
Stack, 301
   Alternative to Stacking, 298
Stacked View. See 3D Stacked View
Stamp Overlays On Image, 195
Standard image intensities, 94
Start IPLab, 29
Status palette
   ROI, 12
Status palette, 27, 31, 42
Subsample, 253
Subtract two images, 258
Subtracting images, 226
Surface plot, 205, 206
Swap 0 and 255, 206
System CLUT, 133, 156
System requirements, 1
  Memory, 3
Tagged Image File Format. See TIFF
Text
  Saving files, 128
  View, 205
Text drawing, 35
Text files, 123
Text view, 125
  Options, 202
Texture, 68
Texture segmentation, 100
Thresholding, 62, 229
TIFF, 127, 128
  Importing from other computers, 159
Tile windows, 41, 301
  Alternative to tiling, 298
Time series analysis, 238
Title, row/column, 195
Toolbars
  FKey Toolbar, 26
Tools
  ROI Tools, 15
Tools Window, 14
Transfer Attributes, 70, 194
Transfer Frames, 192
Transform
  Color, 266
Transforms, 110, 245
Transparent color, 64
Transparent color in the segment layer, 58
Undo, 181
  Pasting, 122
Uneven illumination, 57, 96
Units, 28
  Default, 85
  Distance, 76
  Remembering, 77, 85
  Saving, 128
  Transfer to another image, 77
Units definition
  Distinct names, 77
  Unsigned 16 data, 129
  Unsigned Data Type, 251
  Unsigned<>Signed, 251
  User Data file, 222
  V / # button, 119, 121
Variables, 105, 291
  Action Variables. See Action Variables
  Definition, 119
  Using, 119, 121
Variables, iPLab, 296, 307, 316
Variables, numeric
  Reserved, 342
  Variables window, 28
  Viewing, 28
View As, 93, 125, 205
View Options, 201
Wait for Timer. See Delay
Wand tool, 15, 112
White Balance, 210
Width and height, 31
Window and Leveling, 42
Window list, 301
Windows. See also Files
  Active. See Front Window
  Front. See Front Window
  Image, 9
  Stack, 299, 301
  Tile, 301
XYZ, 133
Zoom. See Magnify tool